UNIVERSIDAD CARLOS III DE MADRID

INGENIERÍA INFORMÁTICA

PROYECTO FIN DE CARRERA

RECOMENDADOR TELEVISIVO SOPORTADO POR BASES DE DATOS RELACIONALES

Autor: Francisco González Doblado

Dirigido por: Elena Castro Galán

Fecha: 25 de Mayo de 2009

Agradecimientos

A mis padres, por la confianza y paciencia que me han ofrecido día a día.

A mis hermanos por su perseverancia, apoyo y ánimos que me han dado.

A mi novia, Eva, por sus ánimos y cariño. Gracias por todas esas ideas brillantes con las que me has ayudado.

A Elena Castro, por todo el tiempo que me ha dedicado, su entusiasmo, interés. Es una suerte contar con una tutora que se muestra tal y como es. Gracias por tu sinceridad.

Índice

Índice de figuras

Índice de tablas

Lista De Acrónimos

- DAO: Data Access Object.
- DTD: Document Type Definition.
- HTTP: HyperText Transfer Protocol.
- JDBC: Java Database Connectivity.
- JDK: Java Developer Kit.
- ODBC: Open DataBase Connectivity
- SQL: Structured Query Language.
- XML: Extended Markup Language.
- W3C: World Wide Web Consortium
- VCR: Videocassette Recorder
- EPGs: Electronic Programming Guides
- STB: Set Top Box
- DVD: Digital Versatile Disc
- ITV: Internet Televisión Interactiva
- PVR: Personal Video Recorder

Prólogo

El contenido de esta memoria de proyecto fin de carrera se estructura como sigue:

En el primer capítulo, capítulo de introducción, se enmarca el proyecto justificando su necesidad y exponiendo su objetivo principal.

El capítulo dos trata sobre la definición del sistema, explicando objetivos, la lógica de la aplicación, y una estructura de tablas donde se identifican los requisitos principales de la aplicación.

A continuación, en el tercer capítulo se expone el diseño del sistema, explicando puntos como el modelo de diseño o los componentes que conforman el sistema.

El capítulo cuarto contiene la implementación realizada, así como los recursos tecnológicos utilizados durante esta fase. En el quinto se estudia la eficacia y la eficiencia del sistema.

Todas las conclusiones extraídas de la realización de este proyecto se exponen en el capítulo 6, y por último, se incluyen dos anexos. En el primero se presenta la gestión del proyecto, y en el segundo un manual de la aplicación para que cualquier usuario pueda aprovechar este proyecto.

Capítulo 1 - Introducción

1.1 Visión general del problema

Actualmente, la oferta televisiva existente es cada vez más amplia, tanto que los usuarios de cable y satélite digitales se sienten a menudo sujetos a un exceso de información. Esta situación puede llevar a una sensación de impotencia al no tener acceso a todos los programas potencialmente de interés. El tiempo se ha convertido en nuestro recurso más valioso y cada vez tenemos menos para ver la televisión. Lleva tiempo controlar si vale la pena grabar un determinado programa o no. Si el usuario tiene acceso a 2400 programas al día (como es potencialmente el caso en España), según un estudio publicado en España en 2005 [Verdarguer, X., 2006], y gasta 1 segundo para evaluar cada uno de ellos, acumula 40 minutos sólo para hacer esta evaluación, más el tiempo para impostar posibles grabaciones. Sistemas de filtro y recomendación automáticos pueden devolver al usuario parte del tiempo gastado en tareas de evaluación. La clave de sus expectativas es cómo de rápido y de bien satisface cualquier compañía el ansia de información que se puede utilizar y en la que se puede confiar. El nuevo espectador quiere recibir la información adecuada en el momento adecuado. Y quien dice información, dice también entretenimiento. Todo empezó con el aparato de vídeo, en 1975, cuando Sony introdujo la noción de time-shift [Ceccaroni, J., 2005]. Hoy el time-shifting ha progresado hasta el punto que millones de usuarios confían no en un VCR (Videocassette Recorder), sino en un sistema de vídeo-grabación digital, que facilita encontrar cualquier cosa en los cientos de canales a disposición, y verla en cualquier momento sin los anuncios. Si la grabación de video proporciona al usuario la habilidad de saltar los anuncios, la confusión y baja calidad de la oferta televisiva actual le proporcionan la razón para hacerlo. La introducción de guías electrónicas de programación de televisión (EPGs, Electronic Programming Guides) y de los grabadores de video pone al alcance del usuario un nuevo tipo de control sobre la programación televisiva. Las EPGs deberían recomendar listas personalizadas de programas y, por otro lado, deberían también estar integradas a fondo en el aparato donde se ve la televisión, con el fin de facilitar el acceso a la colección de archivos digitales del usuario. Los sistemas de gestión, búsqueda y recomendación de la oferta televisiva deberían aprender a conocer a fondo al usuario, para proponerle siempre el programa más adecuado a sus necesidades.

1.2 Antecedentes

La gestión de la nueva generación de televisión está cambiando. Está experimentando una pérdida de identidad como producto a favor de una posición de servicio integral. A nivel de hardware y software, la tecnología DVR (Digital Video Recorder) puede basarse en diferentes configuraciones y, aunque en principio la potencialidad sea la misma, los productos en el mercado presentan diferencias importantes. Las STB (Set top Box,) son un tipo de PVR (Personal video recorder) integradas, con video bajo demanda, que se caracterizan por un coste generalmente más bajo, y por llevar un procesador poco potente y un sistema operativo poco flexible (ej.: Windows CE + Microsoft TV[1]). Las plataformas PC, en cambio, se caracterizan sobre todo por tener grabadora de DVD (Digital Versatile Disc), por una completa integración con Internet y por su expansibilidad. Nuevos programas de gestión de iTV (Internet Televisión Interactiva) aparecen de sorpresa cada mes; ya no es posible identificarlos y monitorearlos todos. A título de ejemplo, se pueden destacar los siguientes servicios:

- Proyecto iMEDIA[2] El usuario puede elegir ver anuncios personalizados e interactivos. El sistema puede gestionar las preferencias televisivas de un grupo familiar.
- Mayordomo de InOutTV[3] Es un agente que revisa gran parte de la programación española, de forma constante, y reconoce en ella todo cuanto se ajusta a las preferencias explícitas del usuario para grabarlo.
- MythTV[4]. Servicio DVR open-source con funcionalidad completa, para sistemas Linux.
- SmartNavigator de Predictive Networks[5]. Es un agente que revisa la programación de compañías de televisión por cable en Norteamérica, de

[1] [http://www.microsoft.com/tv]

[2] [http://imedia.intranet.gr/].

[3] [http://www.inout.tv].

[4] [http://www.mythtv.org]

[5] [http:// www.predictivenetworks.com/]

forma constante, y reconoce en ella todo cuanto se ajusta a las preferencias explícitas e implícitas del usuario, para personalizar la EPG.

- TV FINDER[6]: TV finder es un sistema de gestión, búsqueda y recomendación personalizada de oferta televisiva orientado primariamente a usuarios, y secundariamente a usuarios de equipos PC con funcionalidad de media center. Este sistema incluye los siguientes elementos:
  - Un modelo del dominio con semántica completamente formal (creación y uso de una nueva ontología como sistema de representación del conocimiento).
  - Técnicas de recomendación aplicadas a guías electrónicas de programación de televisión.
  - Un motor de inferencia basado en lógica difusa (uso de descriptores típicos de la experiencia humana).
  - El uso de un sistema multi-agente, que incluye agentes especializados en ejecución autónoma de tareas.
- AVATAR: En el campo de la TV, los sistemas recomendadores pueden permitir al usuario jugar un papel activo y buscar contenidos concretos o, por el contrario, pueden analizar las preferencias de los usuarios, así como los programas que éstos han visto, con el fin de recomendar contenidos personalizados sin una solicitud explícita previa. Esta última posibilidad pretende reducir la implicación de los usuarios en procesos de búsqueda tediosos. AVATAR explora ambas posibilidades. Propone una arquitectura abierta que:
  - Permite actualizar los módulos que generan recomendaciones y añadir nuevos módulos que calculen sugerencias siguiendo diferentes estrategias.
  - Soporta mecanismos para combinar varias recomendaciones, combina un modelo bayesiano y técnicas de razonamiento semántico, una

[6] [ttp://www.tmtfactory.com/articulos/articuloTVFinder200310.pdf]

metodología de inferencia usual en el campo de la Web Semántica, pero novedosa en el campo de los recomendadores televisivos.

La información sobre los usuarios que necesita el re- comendador incluye, al menos, datos personales objetivos, sus preferencias en relación con los contenidos televisivos y registros históricos que almacenen los programas vistos en el pasado. Además de los datos proporcionados por los usuarios, AVATAR debe tener en cuenta el éxito obtenido en recomendaciones previas con el fin de mejorar la calidad de las sugerencias futuras. Esta información de realimentación permite conseguir una rápida convergencia entre las preferencias del usuario y las recomendaciones realizadas.

1.3 Objetivos

El principal objetivo es poder facilitar la elección televisiva en la amplia parrilla que diariamente se oferta, mediante un programa capaz de aconsejar a un usuario que es lo que podría gustarle, pudiendo además retroalimentarse con la elección del propio usuario, para mejorar las futuras recomendaciones.

El objetivo de este trabajo es poder conseguir que los usuarios se sientan cómodos a la hora de ver la televisión.

Para lograr esto, es necesario que el sistema vaya aprendiendo y retroalimentando los gustos del usuario con el que interactúa, y cuanta más información aprenda del usuario, mejores recomendaciones podrá ofrecerle a éste. Pudiendo sugerir día a día las recomendaciones para cada jornada.

Capítulo 2 - Definición del sistema

2.1 Estudios previos

Al inicio del estudio de los objetivos del proyecto, se pensó en modelar el comportamiento de los usuarios, de manera que fuera posible inferir conocimiento sobre su comportamiento. Estudiando el perfil de éstos, mediante su edad, gustos acerca del género, actores, directores y subgéneros del mundo cinematográfico.

De esta manera, basándonos en consumos televisivos anteriores al usuario, y consumos televisivos de usuarios del mismo rango de edad del usuario sometido a estudio, podemos conocer cuales son los gustos que se ajustan a un determinado usuario.

Para inferir toda esta información, nos basamos en una base de datos, cuyo contenido se basa en una serie de tablas que relacionan a los clientes de un proveedor televisivo con los paquetes televisivos que tiene contratado, y qué tipo de programas consume dicho usuario.

A la hora de diseñar el data mart donde almacenaremos la información, es importante que analicemos la dimensión temporal del modelo. Con el fin de inferir conocimiento sobre el comportamiento de los usuarios, nos interesará almacenar en nuestra base de datos no sólo el estado actual del cliente, sino también los cambios en el tiempo que ha sufrido el cliente.

Por ejemplo: Imaginemos un cliente que en enero tenía contratado sólo telefonía, pero que posteriormente se da de alta en TV con el paquete familiar y en marzo acaba de volver al servicio de sólo telefonía. El modelo de usuario puede que coloque a este usuario en otro grupo distinto de los usuarios que no contrataron nunca TV aunque su situación actual es idéntica.

Aunque el modelo de usuario elegido no genera pérdida de información, deberemos guardar todos los datos con los que se ha alimentado el modelo de manera independiente, esto será necesario ya que al estar en fase de experimentación, es probable que tengamos que rehacer el modelo varias veces, por lo que será útil disponer del 100% de los datos en todo momento. Esto generará sobre el sistema redundancia controlada de información.

A continuación mostramos un esquema de tablas donde se relaciona toda esta información:

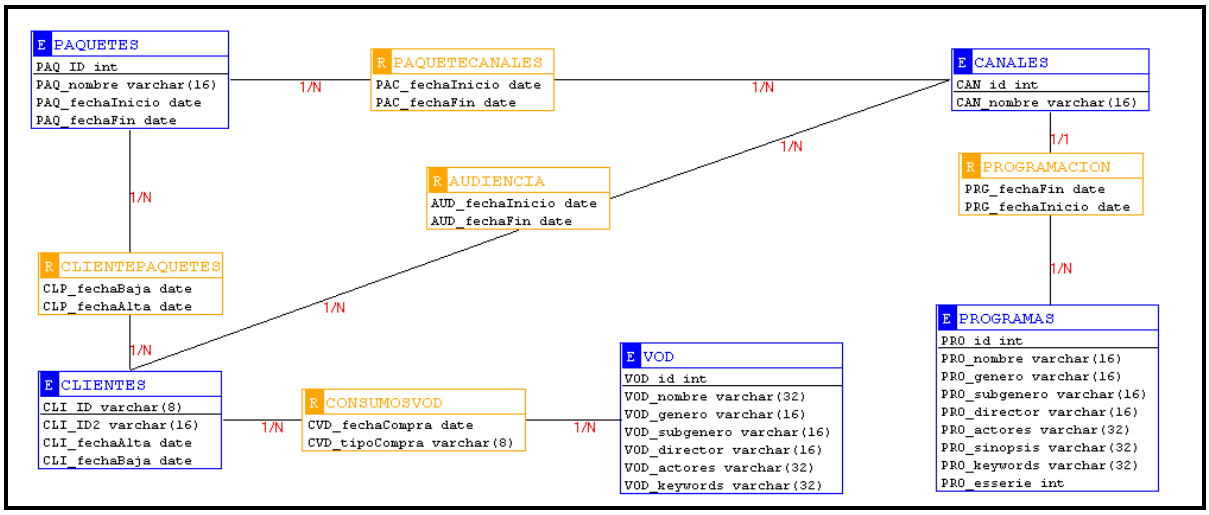


Figura 1 Esquema de tablas data-mart

Con toda la información almacenada de los consumos históricos de los usuarios, podremos inferir que tipo de programas y consumos televisivos le gustarán a otro usuario del mismo rango de edad, o incluso que tipo de programación poder recomendar a un usuario, atendiendo a lo que ha estado viendo en las últimas fechas.

Para poder estudiar el modelado de usuarios, se desarrolló una aplicación basada en una interfaz que permite seleccionar que tipo de gustos tiene un determinado usuario, atendiendo al género, subgénero, actores y directores de los programas y películas de una determinada parrilla televisiva, o incluso cuales de estos gustos tiene un determinado grupo de usuarios asociados a una fecha de nacimiento.

La aplicación permite conocer los gustos del usuario sobre Actores, Directores, Genero y subgénero, o sobre todos estos datos a la vez, combinándolo con la dimensión temporal, el rango de edad, etc…

A continuación se muestran las pantallas de esta aplicación.

La primera pantalla, figura 2, se compone de 2 elementos desplegables, uno seleccionar la información que deseamos saber, como puede ser los Actores, Directores, Genero y Subgénero o bien Genero, Subgénero y Actores. Y otro desplegable para tener en cuenta el año de nacimiento del perfil del usuario para el que deseamos saber esta información, o para todos los usuarios en general, sin tener en cuenta ningún rango de edad. A la derecha de este último desplegable, contamos con un cuadro numérico

donde, en caso de haber seleccionado una determinada edad en el segundo desplegable, poder dar valor numérico a ésta.

Para poder ejecutar la aplicación, simplemente debemos hacer clic sobre el botón buscar, y a continuación, nos aparecerá en el resto de la ventana, la información relativa a la consulta que estamos realizando.

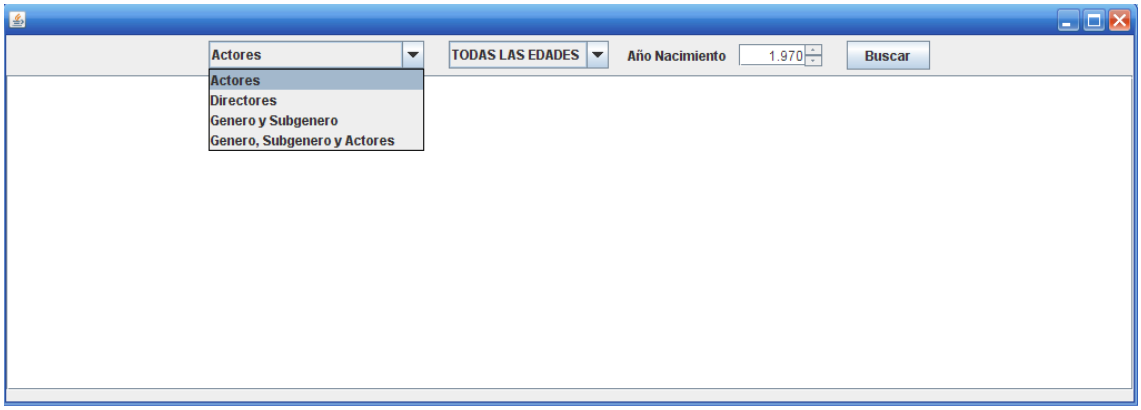


Figura 2 Pantalla inicial perfil usuario.

La figura 2 muestra como en un desplegable, se puede elegir que información acerca del modelado de usuario se desea conocer:

Si elegimos información acerca de los "Actores" y pulsamos "Buscar", podemos ver en la pantalla todos los actores que encajan con el modelo de usuario que estamos estudiando:

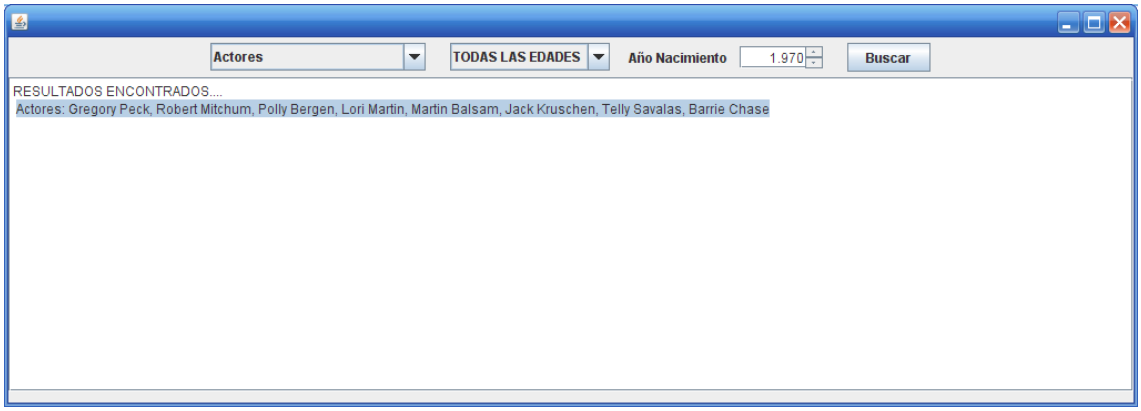


Figura 3 Pantalla final perfil usuario

De la misma forma, lo podríamos haber hecho para directores….

Como se comentó anteriormente, existe la posibilidad de conocer toda esta información, para un grupo de usuarios, dependiendo de su fecha de nacimiento. De esta manera se "inferirá" la información del modelo de un usuario, a partir de los consumos televisivos de otras personas que cumple la misma etapa de edad.

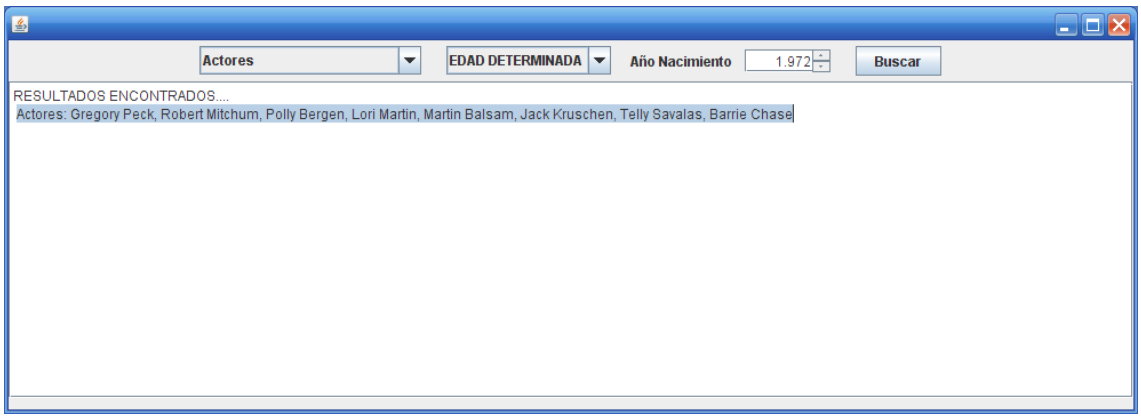


Figura 4 Pantalla final perfil usuario con edad.

2.2 Desglose de objetivos

El objetivo de este trabajo es poder conseguir que los usuarios se sientan cómodos a la hora de ver la televisión.

Para lograr esto, es necesario que el sistema vaya aprendiendo y retroalimentando los gustos del usuario con el que interactúa, y cuanta más información aprenda del usuario, mejores recomendaciones podrá ofrecerle a éste. Pudiendo sugerir día a día las recomendaciones para cada jornada.

En un primer momento, se diseñó un sistema basado en la recomendación de películas, dependiendo de los gustos del usuario, basados en edad, género, actores y directores, que buscaba películas en una base de datos, y devolvía aquellas que mas se ajustaban con los gustos de un usuario.

Para profundizar más en el tema de la recomendación televisiva orientada a usuarios de televisión por cable, se diseñó una aplicación que, mediante las diferentes características del perfil de un usuario, basado en sus gustos sobre películas, géneros,

actores…etc, denominadas keywords, y apoyándonos en una base de datos de sinónimos, hiperónimos y merónimos de palabras como es Wordnet, recoge multitud de palabras similares a las keywords originales del usuario inicial, infiriendo así nuevas cualidades y reuniendo de esta manera un mayor conjunto de atributos asociados al usuario con el que se esta trabajando, consiguiendo así mayor información acerca de las características de un usuario, y poder ofrecer recomendaciones acordes a cada persona.

El perfil de un determinado usuario (keywords), y las keywords "en promoción", son obtenidas de una base de datos, en la cual se mantiene una relación entre el código del decodificador de un determinado usuario, con las keywords relativas a éste. De esta manera, el inicio de una recomendación, comienza con el id del decodificador de un usuario, obteniendo a continuación las keywords relacionadas a éste mediante la base de datos.

2.3 Lógica de la aplicación

Se propone una arquitectura en tres capas, consiguiendo así una modularización del sistema, permitiendo así una independencia de cada módulo y consiguiendo su funcionalidad completa entre distintas arquitecturas.

Nuestro sistema hará las funciones de un decodificador, que interactúe entre el usuario y el operador televisivo, de manera que éste ofrezca al usuario las mejores recomendaciones posibles en un momento dado, en función del perfil de usuario con el que está interactuando en ese momento, y el historial de programación más reciente.

A continuación se muestra un esquema a modo de ejemplo (figura 5), en el cual se puede observar como, mediante un archivo xml, nos llega la petición de recomendación para un determinado usuario, identificado mediante su "id". A continuación se realizará una consulta a la base de datos para recoger todas sus keywords relacionadas, y junto con las keywords de las películas disponibles de recomendación obtenidas en otro archivo xml, se creará una nueva petición en formato xml, que contendrá las keywords del usuario, y las keywords de las películas disponibles en ese momento.

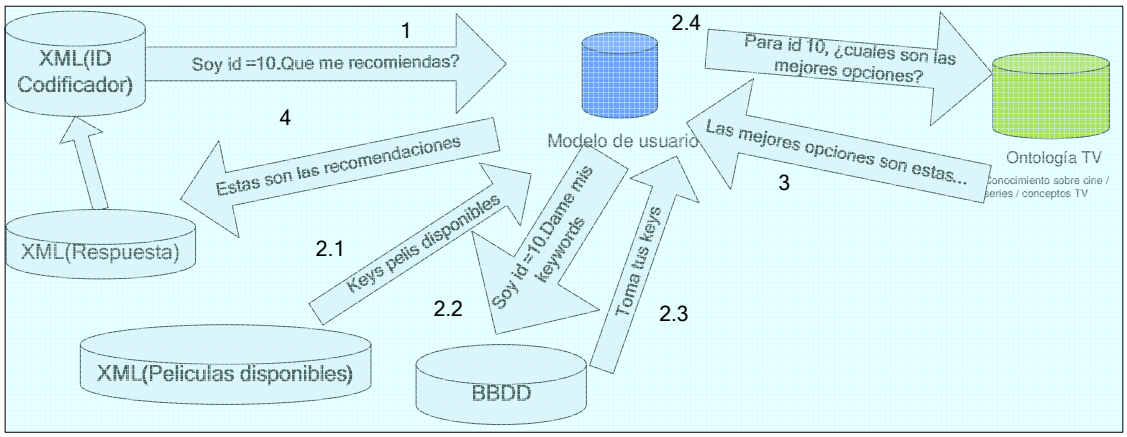


Figura 5 Proceso de recomendación.

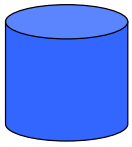
Mediante el modelo de usuario, donde se encuentra almacenado el historial mas reciente de los consumos del usuario, y apoyándonos en la ontología televisiva, donde podremos relacionar similitudes entre keywords, generaremos una recomendación en el mismo formato xml en la que nos llego la petición inicial.

En la siguiente imagen (Figura 6), podemos ver como será la petición inicial, en formato XML SOAP:

Software empotrado de peticiones

1: XML SOAP → Soy el ID=1000 ¿qué me recomiendas?

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
            <soap:Body>
                        <getRecomendados xmlns="http://modelodeusuario.com/recomendados">
                                    <decodificadorID>1000</decodificadorID>
                        </getRecomendados>
            </soap:Body>
</soap:Envelope>
```



Modelo de usuario

Acepta peticiones concurrentes

2: XML SOAP → ¿Cómo encaja el programa A con los que programan que emiten hoy (B,C y D)?

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <getEncajeSinopsis xmlns="http://ontología.com/encaje">
      <keywordsFuente id="A">blanco,negro,comedia,pianista</keywordsFuente>
      <keywordsDestino id="B">noche,opera,comedia,harpo,groucho,risas< keywordsDestino>
      <keywordsDestino id="C">Humphrey,Bogart,paris,pianista,color<keywordsDestino>
      <keywordsDestino id="D">naufrago,isla,tom,cocos,accidente,avion< keywordsDestino>
    </getEncajeSinopsis>
  </soap:Body>
</soap:Envelope>
```

Figura 6 Xml con petición del usuario.

En la figura 6 podemos ver como se podrían componer las peticiones que viajan entre el decodificador y el modelo de usuario y posteriormente entre el modelo de usuario y la ontología.

En el recuadro 1 de la figura 6, podemos ver como dentro de la etiqueta <decodificadorID> el decodificador informaría de su identificador y preguntaría por los recomendados para hoy. El modelo de usuario verificaría el ID del decodificador y lo clasificaría; una vez clasificado y en función de sus gustos, seleccionaría los programas que se están emitiendo hoy y se los pasaría a la Ontología para que le devolviese las distancias entre los títulos.

De esta manera, se crearía el fichero xml del recuadro 2 de la figura 6, donde dentro de la etiqueta <keywordsFuente> se clasificarían los gustos del usuario identificado con el id del decodificador que ha llegado en la petición, y dentro de la etiqueta <keywordsDestino> se inferirán las características de cada una de las películas que se están emitiendo hoy, para que ahora la Ontología las evalúe y decida cuales son las que mas se ajustan al usuario.

Con toda esa información el modelo de usuario estaría en disposición de inferir una respuesta y devolvérsela al decodificador.

El siguiente paso por tanto, es la creación de un nuevo fichero xml con la respuesta inferida, con las películas recomendadas para el usuario, y que ésta llegue al decodificador.

En la figura 7, se propone una estructura XML de respuesta de la ontología de películas con el modelo de usuario. En el recuadro 3 de la figura, se ordenan ascendentemente, según la afinidad que tenga la película hacia el usuario, todas las películas recomendadas para hoy. El orden de preferencia viene designado con el atributo “orden”, dentro de la etiqueta <recomendado> y el código de la película que se recomienda en cada caso.

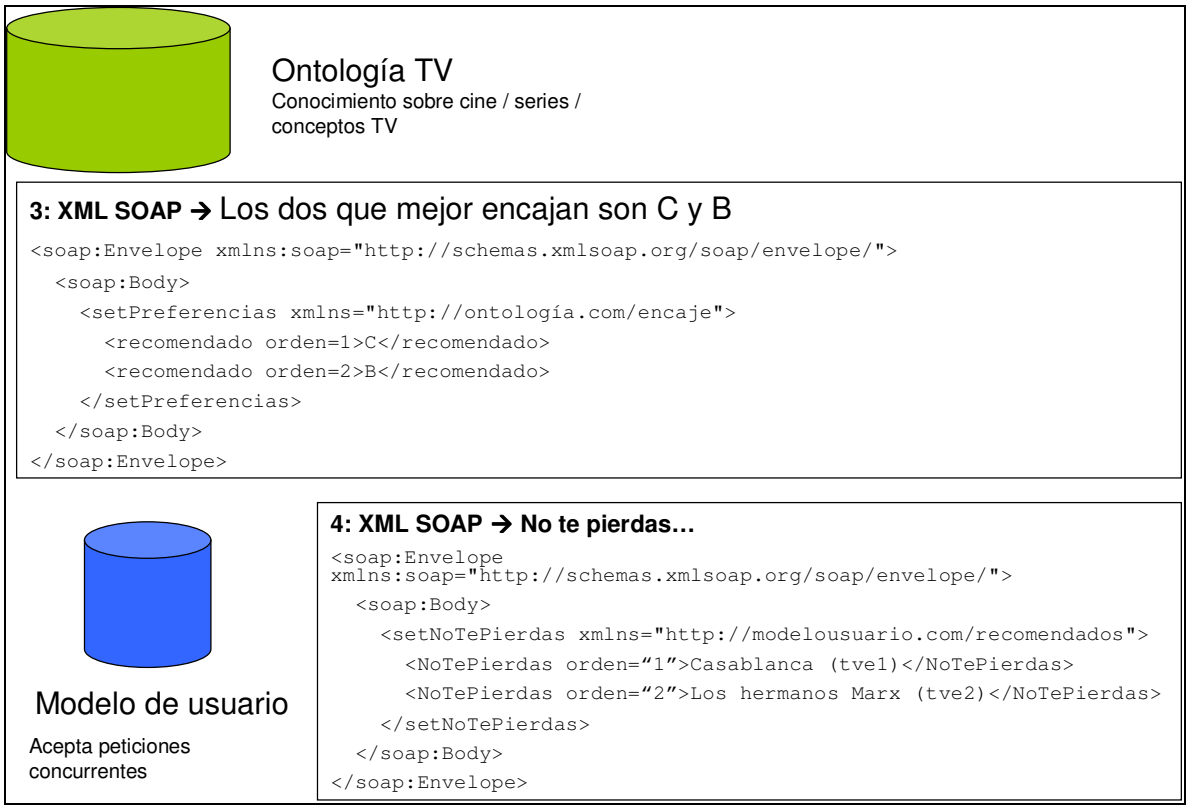


Figura 7 Propuesta de intercambio de mensajes con el servidor**.**

A continuación en el recuadro 4 de la figura 7, se crea el fichero xml de respuesta final, donde se consultan las características de las películas recomendadas en el recuadro 3, pero incluyendo en este nuevo fichero las características de la película, tales como titulo, cadena en que se emite, etc…

Este tipo de comunicación utilizando un estándar como XML SOAP ayudaría a poder trabajar con distintas plataformas de modo que cada módulo del sistema pudiese estar soportado en arquitecturas y lenguajes diferentes.

En resumen, una modularización como la que se propone, deja el sistema preparado para el futuro, ya que utiliza un interfaz de comunicación que no depende de la tecnología y, por ello, dota al sistema de un mayor nivel de reutilización, autonomía, escalabilidad y tolerancia a fallos (se pueden aislar los módulos y el fallo de uno no afecta al resto del sistema que puede continuar dando servicio)

2.4 Obtención de requisitos

Por la naturaleza del sistema, un proyecto orientado a la experimentación, la toma de requisitos inicial, es muy probable que sufra modificaciones a lo largo del ciclo de vida del proyecto, con el fin de enfocar mejor la experimentación final del sistema.

Se van a especificar las capacidades y restricciones del software que va a ser usado, cuáles son las ventajas u objetivos de su uso, siempre visto desde el punto de vista del usuario, con un lenguaje y una terminología que resulten claros y cercanos al mismo.

La meta que se quiere alcanzar es obtener una visión clara de cuáles son las necesidades del cliente.

La especificación de requisitos se va a hacer siguiendo la metodología propuesta por el estándar de la ESA-Lite[7]

2.4.1 Requisitos específicos

La nomenclatura para los requisitos del sistema será la siguiente. Los requisitos de usuario se clasificarán en dos categorías:

Requisitos de capacidad.

Requisitos de restricción.

Cada requisito tiene un identificador unívoco, el cual servirá para poder ser identificado a lo largo de todos los documentos del proyecto. Dicho identificador tendrá los siguientes formatos:

RUC-XXX

RUR-XXX

El primer formato (RUC-XXX) será el usado para los requisitos de capacidad. Mientras que el segundo (RUR-XXX) será el usado para identificar los requisitos de restricción.

En los dos casos XXX son tres dígitos decimales que permiten enumerar de forma secuencial los requisitos.

[7] Estándar ESA Documento E.S.A Lite en Español

Por ejemplo: RUC-001 será el primer requisito de capacidad.

2.4.2 Requisitos de Capacidad

IDENTIFICADOR	RUC-001
NOMBRE	**Interfaz de usuario simple y auto contenida.**
DESCRIPCIÓN	La interfaz de usuario debe permitir en todo momento, tanto el proceso que se esta realzando, como el progreso en curso de éste.
NECESIDAD	☑ Esencial ☐ Conveniente ☐ Opcional
ESTABILIDAD	☑ Estable ☐ No estable
PRIORIDAD	☑ Alta ☐ Media ☐ Baja

Tabla 26 - Requsito de capacidad 001.

IDENTIFICADOR	RUC-002
NOMBRE	**Inicio de cada subproceso**
DESCRIPCIÓN	El sistema debe proporcionar un botón “Stara” para que el usuario pueda iniciar cada subproceso del sistema.
NECESIDAD	☑ Esencial ☐ Conveniente ☐ Opcional
ESTABILIDAD	☑ Estable ☐ No estable
PRIORIDAD	☑ Alta ☐ Media ☐ Baja

Tabla 27 - Requsito de capacidad 002.

IDENTIFICADOR	RUC-003
NOMBRE	**Manual de usuario**
DESCRIPCIÓN	Se dispondrá de un manual de usuario accesible en todo momento por los usuarios de la aplicación.
NECESIDAD	☑ Esencial ☐ Conveniente ☐ Opcional

ESTABILIDAD	☑ Estable ☐ No estable
PRIORIDAD	☐ Alta ☐ Media ☑ Baja

Tabla 28 - Requsito de capacidad 003

IDENTIFICADOR	RUC-004
NOMBRE	**Inferencia de datos**
DESCRIPCIÓN	Dados algunos datos "keywords" relativos a una película, el sistema será capaz de inferir conocimiento a partir de éstos datos, obteniendo de esta manera un mayor número de éstas.
NECESIDAD	☑ Esencial ☐ Conveniente ☐ Opcional
ESTABILIDAD	☑ Estable ☐ No estable
PRIORIDAD	☐ Alta ☐ Media ☑ Baja

Tabla 29 - Requsito de capacidad 004.

2.4.3 Requisitos de Restricción

IDENTIFICADOR	RUR-001
NOMBRE	**Lenguaje de desarrollo**
DESCRIPCIÓN	El lenguaje utilizado para el desarrollo de la aplicación será Java, sobre el entorno de desarrollo eclipse 3.1.2.
NECESIDAD	☑ Esencial ☐ Conveniente ☐ Opcional
ESTABILIDAD	☑ Estable ☐ No estable
PRIORIDAD	☑ Alta ☐ Media ☐ Baja

Tabla 30 - Requsito de restriccion 001.

IDENTIFICADOR	RUR-002
NOMBRE	**Funcionamiento del prototipo**

DESCRIPCIÓN	El prototipo deberá funcionar en cualquier sistema hardware y software con una versión de la máquina virtual de java 1.5.0_02 o superior.
NECESIDAD	☑ Esencial ☐ Conveniente ☐ Opcional
ESTABILIDAD	☑ Estable ☐ No estable
PRIORIDAD	☑ Alta ☐ Media ☐ Baja

Tabla 31 - Requsito de restriccion 002.

IDENTIFICADOR	RUR-003
NOMBRE	**Interfaz sencilla e intuitiva**
DESCRIPCIÓN	Los elementos de la interfaz de usuario estarán dispuestos de tal forma que el uso de la misma resulte fácil e intuitivo.
NECESIDAD	☑ Esencial ☐ Conveniente ☐ Opcional
ESTABILIDAD	☑ Estable ☐ No estable
PRIORIDAD	☑ Alta ☐ Media ☐ Baja

Tabla 32 - Requsito de restriccion 003.

IDENTIFICADOR	RUR-004
NOMBRE	**Interacción con el usuario**
DESCRIPCIÓN	El usuario podrá interactuar con el sistema mediante el uso de ratón.
NECESIDAD	☑ Esencial ☐ Conveniente ☐ Opcional
ESTABILIDAD	☑ Estable ☐ No estable
PRIORIDAD	☑ Alta ☐ Media ☐ Baja

Tabla 33 - Requsito de restriccion 004.

Capítulo 3 - Diseño de componentes del sistema

3.1 Método de diseño

En este punto, se van a detallar las especificaciones tomadas para definir el modelo arquitectónico del sistema. La primera consideración a tener en cuenta, es que según las especificaciones, existe una división física entre los diferentes módulos. En este sentido, y dado el carácter distribuido del sistema, se parte para el diseño de modelos de arquitecturas basadas en componentes independientes, que se comunican entre sí para intercambiarse información, terminando toda la información final en la interfaz, a partir de la cual podremos visualizar las películas que mas le interesen a un usuario.

3.2 Descripción de la descomposición

En este apartado se describen los diferentes componentes que forman la aplicación de recomendación de películas.

El sistema se descompone en módulos o componentes, para poder abstraer cada parte independiente de la aplicación, con el objetivo de conseguir un sistema flexible a posibles cambios o mejoras. De esta manera podríamos tener que realizar una modificación en una funcionalidad de la aplicación, y éste cambio en cualquiera de los módulos, no tendría que afectar en el funcionamiento de los demás, haciéndolo lo más robusto y dinámico posible.

En este modelo de descomposición, podemos diferenciar cinco componentes, cada uno de ellos se comunica con los demás para intercambiarse información.

A continuación, en la figura 8 se muestra la descomposición de cada uno de los paquetes que se han especificado para formar el sistema.

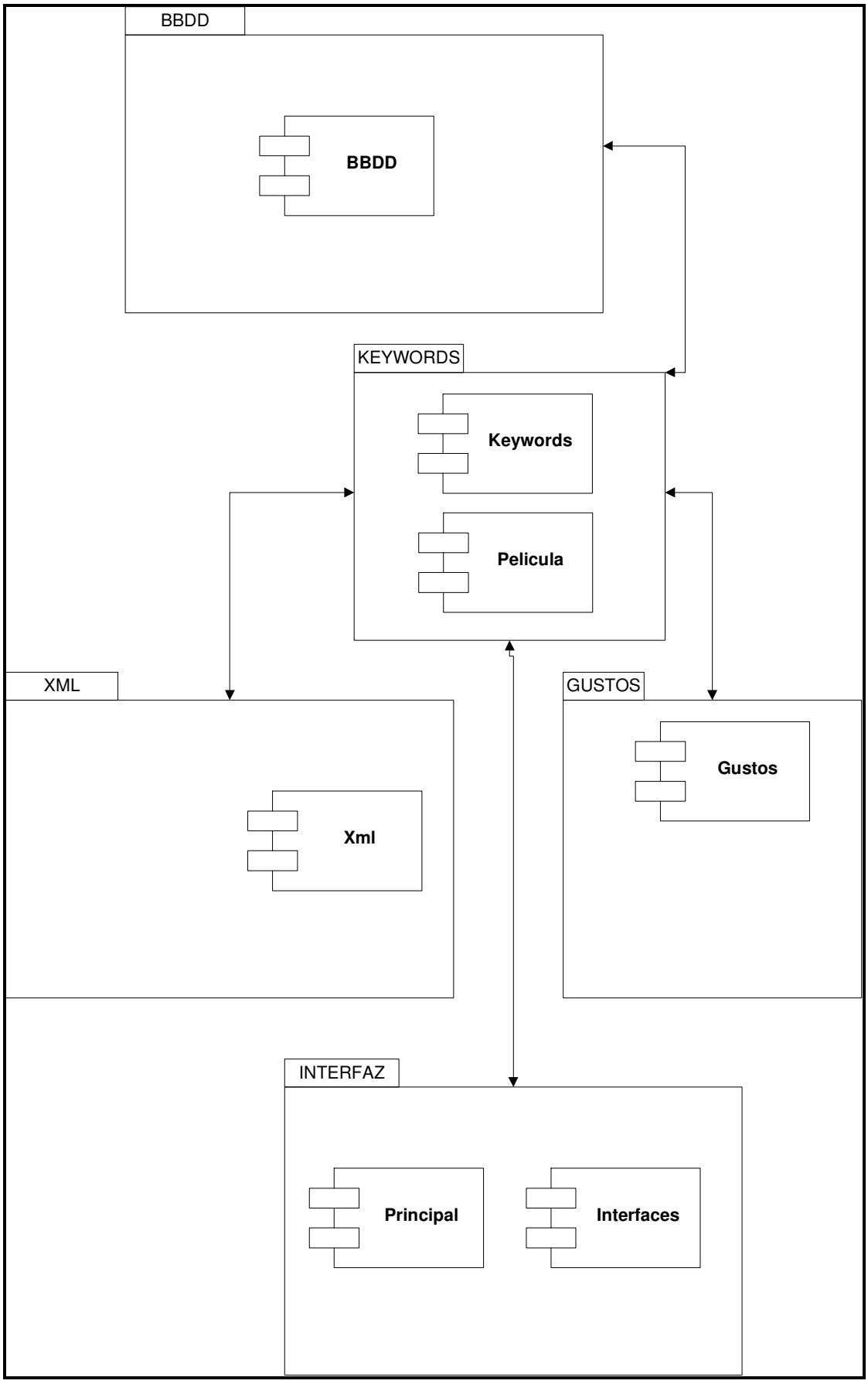


Figura 8 Diagrama de componentes.

3.3 Descripción de los componentes

Se describen a continuación los componentes del modelo.

Cada componente está identificado mediante el nombre del paquete que tiene asignado en la implementación de la aplicación.

3.3.1 Componente Base de datos

TIPO	Paquete java.
FUNCIÓN	La función del componente es implementar el código necesario para la lectura de la base de datos.
INTERFACES	Como salida, retorna los datos relativos a un modelo de usuario.
DATOS	Datos almacenados.
RECURSOS	Es necesario tener la librería que permite leer y conectarse con la base de datos de Oracle.

Tabla 34 - Componente Base de datos.

3.3.2 Componente Gustos

TIPO	PAQUETE JAVA.
FUNCIÓN	LA FUNCIÓN DEL COMPONENTE ES IMPLEMENTAR EL CÓDIGO NECESARIO PARA PODER REALIZAR EL MODELADO DE UN USUARIO, A PARTIR DE LOS CONSUMOS TELEVISIVOS REALZADOS.
INTERFACES	A INTERFAZ ES A TRAVÉS DE LA VENTANA DE LA APLICACIÓN.
DEPENDENCIAS	LA DEPENDENCIA ESTA ESTABLECIDA CON EL COMPONENTE "PRINCIPAL" QUE ES EL QUE EJECUTA LA APLICACIÓN Y CON EL COMPONENTE "INTERFACES" QUE ES EL QUE VISUALIZA LA APLICACIÓN POR PANTALLA.
PROCESAMIENTO	EL COMPONENTE ACCEDERÁ A LOS DATOS ALMACENADOS EN LA BASE DE DATOS SOBRE LOS CONSUMOS DE LOS USUARIOS.
DATOS	NO PROCEDE.

Recursos	No procede.

Tabla 35 - Componente Gustos.

3.3.3 Componente Interfaces

Tipo	Paquete java.
Función	La función del componente es implementar la ventanas, gráficos, eventos de interacción, controles y animaciones que forman la interfaz del sistema.
Interfaces	La interfaz es a través de la interacción del usuario con los elementos en pantalla.
Procesamiento	El componente mostrará todos los elementos gráficos y quedará a la espera de eventos de interacción con el usuario.
Datos	No procede.
Recursos	No procede.

Tabla 36 - Componente Interfaces.

3.3.4 Componente Keywords

Tipo	Paquete java.
Función	La función del componente es la de almacenar todas las keywords de un usuario y de cada una de las películas
Interfaces	Se comunica con el componente "película" y con el componente "bbdd".
Dependencias	Depende del componente películas
Procesamiento	El componente recibirá mensajes del componente película para almacenar todas las keywords de las películas y del usuario.
Datos	No procede.
Recursos	No procede.

Tabla 37 - Componente Keywords.

3.3.5 Componente Película

TIPO	PAQUETE JAVA.
FUNCIÓN	LA FUNCIÓN DE ESTE COMPONENTE ES LA DE COMPARAR LAS KEYWORDS DE LAS PELÍCULAS CON LAS KEYWORDS DEL USUARIO, PARA PODER DETERMINAR CUALES SON LAS MÁS RECOMENDADAS, Y ORDENARLAS SEGÚN SU AFINIDAD CON EL USUARIO.
INTERFACES	DEFINE EL CONJUNTO DE FUNCIONES QUE COMPARAR LAS SIMILITUDES EXISTENTES ENTRE LAS KEYWORDS DE UNA PELÍCULA Y LAS KEYWORDS DE UN USUARIO.
DEPENDENCIAS	DEPENDE DEL COMPONENTE KEYWORDS.
DATOS	NO PROCEDE.
RECURSOS	NO PROCEDE.

Tabla 38 - Componente Película.

3.3.6 Componente Xml

TIPO	PAQUETE JAVA.
FUNCIÓN	LA FUNCIÓN DEL COMPONENTE ES IMPLEMENTAR EL CÓDIGO NECESARIO PARA LA LECTURA DE FICHEROS XML DE ENTRADA, COMO EL ID_USUARIO Y EL FICHERO DE PELÍCULAS DISPONIBLES, Y LA CREACIÓN DE FICHEROS XML DE SALIDA, COMO LAS PELÍCULAS RECOMENDADAS.
INTERFACES	LA INTERFAZ DE COMUNICACIÓN ESTA FORMADA POR EL CONJUNTO DE FUNCIONES QUE PERMITEN LEER UN FICHERO XML Y OBTENER LOS DATOS
DEPENDENCIAS	NO PROCEDE.
PROCESAMIENTO	EL COMPONENTE ACCEDERÁ AL ARCHIVO XML CON LA BASE DE DATOS. EL SIGUIENTE PASO SERÁ EXTRAER LA INFORMACIÓN Y CREAR EL FICHERO XML FINAL, CON LAS PELÍCULAS RECOMENDADAS.
DATOS	FICHERO XML CON ID DE USUARIO, FICHERO XML CON PELÍCULAS DISPONIBLES.
RECURSOS	NO PROCEDE.

Tabla 39 - Componente Xml.

3.3.7 Componente Principal

TIPO	PAQUETE JAVA.
FUNCIÓN	SE ENCARGA DE EJECUTAR LA APLICACIÓN, Y MOSTRAR EL PROGRESO DE CADA UNA DE LAS TAREAS DEL PROCESO DE RECOMENDACIÓN
INTERFACES	SE COMUNICA CON EL COMPONENTE INTERFACES.
DEPENDENCIAS	NO PROCEDE
PROCESAMIENTO	EL PAQUETE SE ENCARGARÁ DE EJECUTAR LA APLICACIÓN Y EJECUTAR CADA UNA DE LAS FUNCIONALIDADES DE ÉSTA, EN FUNCIÓN DE LOS ACCIONES REALIZADAS POR EL USUARIO SOBRE EL INTERFAZ GRÁFICA
DATOS	NO PROCEDE.
RECURSOS	NO PROCEDE.

Tabla 40 - Componente Principal.

Capítulo 4 - Implementación del sistema

4.1 Tecnologías utilizadas en la implementación

Las tecnologías en las que se ha basado el desarrollo del sistema son:

- XML y XMLSchemas.
- Java.
- ODBC y JDBC.

A continuación describiremos cada una de estas tecnologías.

4.1.1 XML y XMLSchema

Todo el proyecto se apoya en los estándares XML (Lenguaje de Marcado Extensible = eXtensive Markup Language) y XMLSchema, que con su simplicidad y potencia permiten gestionar información de forma eficiente y controlada. XML es un metalenguaje que permite estructurar la información de un fichero de texto utilizando etiquetas de marcado.

El XML, basado en SGML, permite la codificación para la distribución de documentos complejos por Internet, mediante etiquetas "ad hoc" definidas por el autor de mismo. XML reúne tres condiciones básicas, debido a sus características, y podemos decir que es:

1. Formal: pues permite establecer la validez de los documentos.
2. Estructurado: para que sea capaz de manejar documentos complejos.
3. Ampliable: para facilitar la gestión de grandes depósitos de información.
3. Ampliable: para facilitar la gestión de grandes depósitos de información.

Un ejemplo de como funciona XML sería algo así:

<agenda>
<persona>
<nombre>Kike</nombre>
<telefono>638002993</telefono>
<comentario>Es un bombon</comentario>

```
</persona>
<persona>
<nombre>Maria</nombre>
<telefono>956-78.90.12</telefono>
<telefono>652135792</telefono>
</persona>
</agenda>
```

Junto a la sintaxis de datos del XML existe también lo que se conocen como esquemas, que son como la gramática necesaria para poder expresarse con un lenguaje. Son esquemas que definen si el contenido de un documento XML es válido o no. Los esquemas más conocidos para el XML son:

1. DTD (Document Type Definition): que es un modelo ya un tanto antiguo y con sintaxis más limitada.
2. Esquema XML o XML Schema: más potente, moderno y complejo.

El XML Schema apareció como evolución de la DTD (Document Type Definition), ya que subsana todas las carencias que tenia la DTD, aumentando la funcionalidad de las aplicaciones que utilizan XML.

4.1.2 Java

Se ha utilizado el lenguaje JAVA para realizar toda la implementación de la aplicación.

La elección de este lenguaje se ha debido a que es Orientado a objetos, Distribuido, indiferente de la arquitectura lo cual es muy importante para este proyecto, y muy robusto.

4.1.3 ODBC y JDBC

ODBC (Open DataBase Connectivity) es un estándar que define un método de acceso a bases de datos desde cualquier aplicación, sin importar qué sistema gestor de base de datos aloje la instancia de base de datos [Wikipedia, 2007 c)]. JDBC (Java Database Connectivity) es el API que permite acceder a bases de datos desde el lenguaje de programación Java utilizando lenguaje SQL, de esta manera conseguiremos acceder a toda la información sobre los modelos de usuario y las características de los productos televisivos, almacenados en la base de datos.

4.2 Herramientas utilizadas durante la implementación

- **Eclipse 3.2:** es el entorno de programación utilizado para toda la codificación de la aplicación. Es una de las herramientas mas utilizadas por las empresas de desarrollo de software en el lenguaje Java
- Oracle: como herramienta cliente/servidor para la gestión de las bases de datos.
- Wordnet: como repositorio de datos para almacenar todas las relaciones de sinónimos, hiperónimos e hipónimos de palabras.

4.3 Recursos tecnológicos

Para la realización del proyecto se ha utilizado un ordenador con las siguientes características:

- Procesador: AMD Turion 64 x 2.
- Memoria RAM: 1024 MB (DDR2-667 DDR2 SDRAM).
- Disco Duro: 200 GB.
- Conexión a Internet: 2 Mbps.
- Sistema Operativo: Microsoft Windows XP Professional (Service Pack 2).

4.4 El proceso recomendación

En este punto, se va a introducir a modo resumen, el proceso que se lleva a cabo para poder crear una recomendación televisiva que conlleva 4 pasos desde la petición de la recomendación por parte del cliente, pasando por la inferencia de los gustos y palabra clave del perfil del usuario, hasta llegar a la clasificación de las películas y finalizando con la creación de la respuesta por parte del servidor, que será enviado al cliente.

A continuación se muestran en la imagen (figura 9), los 4 pasos en los que se basa la aplicación para crear una recomendación acorde a cada usuario:

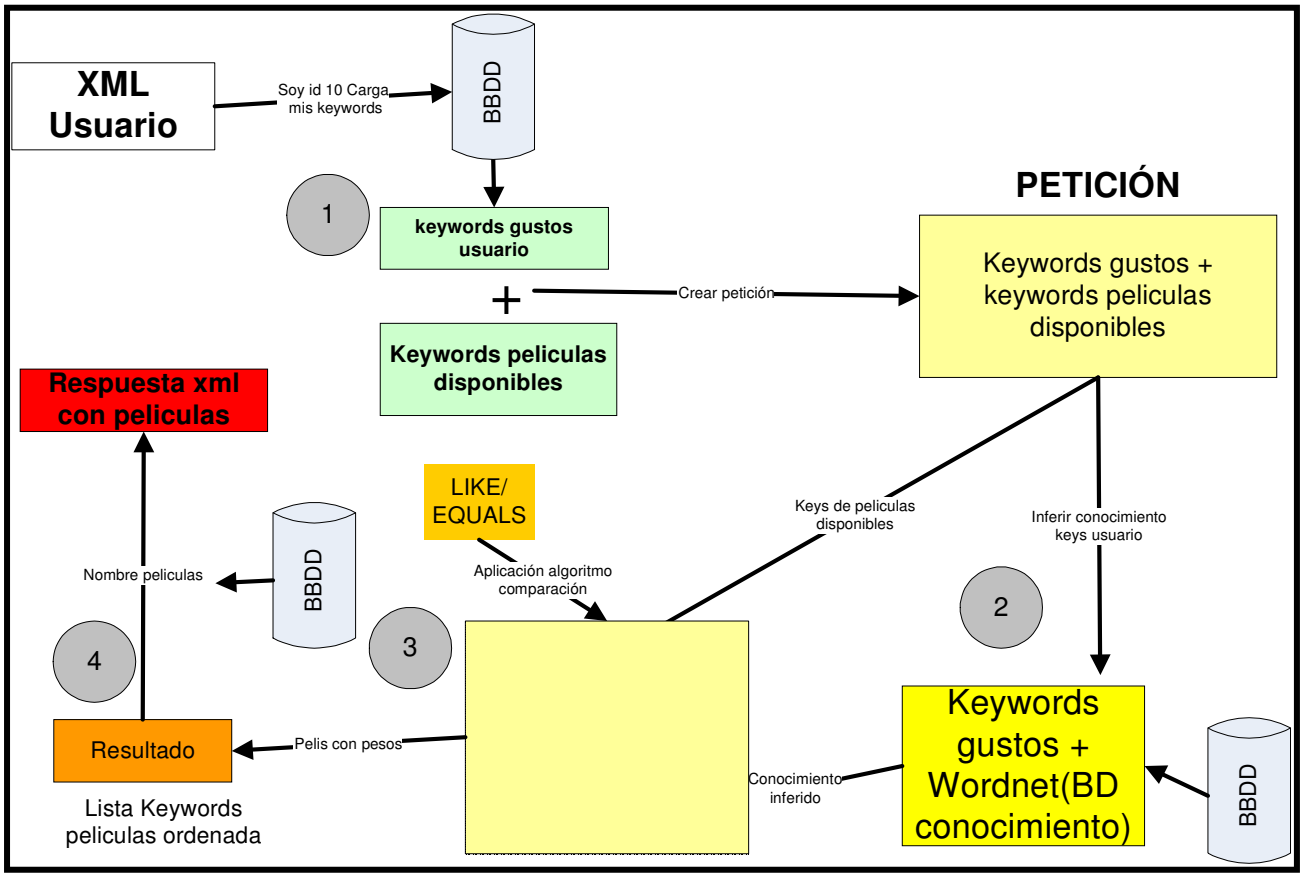


Figura 9. Proceso completo de recomendación.

A continuación se explican brevemente cada uno de los cuatro pasos de la Figura 4, que componen el proceso de recomendación:

Paso1:

El primer paso es seleccionar un fichero xml que corresponderá al id del usuario para el que se va a realizar la recomendación. Así como el algoritmo con el que se desea comparar las keywords del usuario con las keywords de las películas (se dará opción de elección sobre dos algoritmos).

A continuación, se cargarán las keywords asociadas al usuario para el que se esta realizando la recomendación. Para ello, se realizará una consulta a la base de datos, donde esta relacionado de manera directa, un id de usuario, con su correspondiente decodificador, y los gustos (keywords) de éste.

Paso 2:

Una vez obtenido el valor de las keys que caracterizan a un usuario, se creará un fichero xml correspondiente a la petición de recomendación, que incluirá las keywords del usuario obtenidas en el anterior punto, junto con las películas disponibles en la parrilla televisiva en ese mismo momento. Para simular el catálogo de películas disponibles, se utilizará otro fichero xml, llamado películasDisponibles.xml, en el cual incluiremos las películas que queramos ofrecer en cada momento

En el momento en el que hemos cargado las keys del usuario de la base de datos, y hemos obtenido las películas disponibles mediante el fichero peliculasDisponibles.xml, crearemos un fichero xml, llamado "petición.xml", con el contenido de toda esa información, que nos servirá para enviarla al servidor, proporcionándole así todo lo necesario para que nos proporcione la mejor elección posible.

Tras crear el fichero peticion.xml, partiremos de la idea de envío hacia el servidor, donde en el momento de la recepción del mismo, éste extraerá, por un lado, las keywords que definen el perfil del usuario, y por otro las keywords características de cada una de las películas disponibles para la recomendación.

Una vez cargadas todas las keywords, tanto las del usuario, como las de las películas, recorreremos una a una las keywords que componen la lista de keywords de gustos del usuario, buscando en la base de datos, los sinónimos, hiperónimos e hipónimos correspondientes a cada una de ellas

En este momento, tenemos por un lado las listas de keywords de las películas disponibles, y por otro lado, una lista "ampliada" con conocimiento inferido a través de la base de datos de sinónimos, de los gustos del usuario para el que estamos trabajando.

Paso 3:

Llegado a este punto, pasaremos a evaluar cómo de afín es una película para los gustos del usuario. Para ello, recorremos una a una las keywords de cada película, e

iremos comparando cada una de ellas con las keywords de los gustos del usuario, asignando de ésta manera una puntuación significativa a cada película, dependiendo de la similitud de las keyword de la película con las keywords del usuario.

Para realizar la comparación de palabras clave, podremos utilizar dos algoritmos, denotados como “EQUALS “Y “LIKE”, los cuales pasamos a explicar a continuación:

- EQUALS: Este algoritmo viene implementado en el lenguaje Java, y lo que hace es comparar cadenas, carácter a carácter, devolviendo trae si todos los caracteres que conforman la cadena son iguales, y false en caso contrario.

  Ej: ‘casa’ equals ‘casa’ = trae

  ‘casa’ equals ‘casona’ = false

- Por otro lado se ha implementado el algoritmo LIKE, que se basa en la comparación “parcial” de cadenas, intentado comparar básicamente el lexema de las palabras, sin tener en cuenta prefijos, que éstas puedan contemplar. De esta manera tendremos en cuenta la coincidencia de al menos el 50% del inicio de la palabra, para la comparación de ambos. Así, si tomamos el ejemplo utilizado en el punto anterior con el algoritmo Equals: ‘casa’ like ‘casona’ true

El algoritmo que se ha implementado es “LIKE”, a continuación se muestra su código fuente, junto con una breve explicación del mismo:

ALGORITMO LIKE

```
for (int i= 0;i < listaKeywords.size();i++){
        Keyword key = (Keyword)listaKeywords.get(i);
        for (int j= 0;j < pelicula.keywords.size();j++){
        //Le adjudicamos el peso por sinónimos
        for (int h= 0;h < key.sinónimos.size(); h++){
```

Recorremos todas las keywords de la lista

Las comparamos con las keywords de la película

```
char[]lexkeySinonimo=key.sinonimos.get(h).toString().toCharArray();
```

```
char[] lexkeyPelicula = pelicula.keywords.get(j).toString().toCharArray();
```

Comparamos todas las letras de las dos palabras

```
//si coinciden en un 50 % asigno peso de sinónimo
int max = Math.min(lexkeySinonimo.length, lexkeyPelicula.length);
int similar=0;
        for (int fin= 0; fin< max/2; fin++){
                if (lexkeyPelicula[fin] == lexkeySinonimo[fin])
                similar++;
        }
        if (similar>= max/2){
        pelicula.aumentarPesoSinonimo();
                        }
```

Si coinciden mas de la mitad de las letras, concluimos que la palabra es valida, y por lo tanto le asignamos peso.

Paso 4:

Tras asignar a cada película su relevancia correspondiente, se ordenará la lista de películas de manera descendente, atendiendo a la relevancia obtenida, y se elegirán las n primeras para crear la respuesta

Para la creación de la respuesta de recomendación, se generará un fichero xml, con las películas recomendadas, atendiendo a la petición que nos llego al inicio de la ejecución.

Tras crear la respuesta en la parte del servidor, se generará la recomendación al usuario, la cual contendrá los títulos y la información relevante de las películas recomendadas. Para ello, se realizarán consultas a la base de datos donde, a partir de cada id de las películas obtenido en la respuesta, se conseguirán todos los datos relativos a éstas para presentarlos al usuario final.

A continuación se desglosa paso a paso como se produce una recomendación para un usuario:

1. Selección del fichero XML correspondiente al id del usuario para el que se va a realizar la recomendación.
2. Consulta de keywords asociadas al id del usuario anterior.
3. Creación del xml de petición, a partir de las keywords obtenidas de la base de datos del usuario, junto con las películas disponibles que se obtendrán de otro fichero xml (peliculasDisponibles.xml)
4. Extracción de keywords asociadas al usuario y keywords de cada una de las películas disponibles para la recomendación.
5. Inferencia de sinónimos, hiperónimos e hipónimos de cada una de las keywords relacionadas con las películas disponibles.
6. Asignación de pesos, a las diferentes películas, dependiendo de las ocurrencias exactas entre las keywords inferidas de las películas con las relativas a las del usuario.
7. Elección de recomendación, tras la asignación de pesos, se ordenarán las películas según el peso asignado, y se seleccionarán las n-primeras.
8. Creación de respuesta en formato xml, con las películas seleccionadas en el punto anterior.
9. Envío de la recomendación. La respuesta creada en el apartado anterior, se generaría en la parte del servidor, y éste, generará la recomendación que será enviada al cliente.

A continuación se explicarán en detalle cada una de las fases del proceso.

4.4.1 Selección XML id usuario

Debido a que nuestro sistema es local, el primer paso es seleccionar un fichero xml que corresponderá al id del usuario para el que se va a realizar la recomendación. Así como el algoritmo con el que se desea comparar las keywords del usuario con las keywords de las películas (equals ó like).

El xml que contiene el id del decodificador del usuario tiene la siguiente estructura:

<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">

<soap:Body>

<getRecomendados xmlns="http://modelodeusuario.com/recomendados">

<decodificadorID>001</decodificadorID>

</getRecomendados>

</soap:Body>

</soap:Envelope>

En este fichero, podemos destacar las siguientes etiquetas:

- <getRecomendados> : modelo de usuario del cual obtenemos los datos del usuario
- <decodificadorID> : Identificador del codificador que identifica a un usuario.

Sobre este fichero, deberemos recoger el valor de la etiqueta <decodificadorID>. En este caso el id del decodificador será 001, que será el valor que identifique al usuario.

4.4.2 Obtención keywords id usuario

El siguiente paso, será obtener las keywords asociadas al usuario para el que se esta realizando la recomendación. Para ello, se realizará una consulta a la base de datos, donde esta relacionado de manera directa, un id de usuario, con su correspondiente decodificador, y los gustos (keywords) de éste.

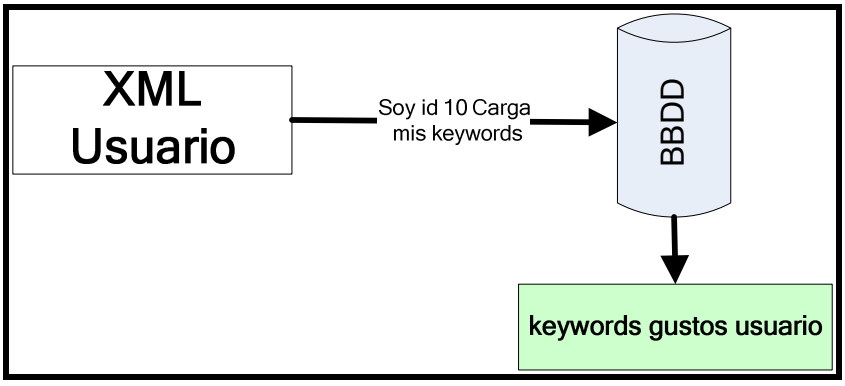


Figura 10 Carga de keywords del usuario

Al acceder a la base de datos podemos ver que el decodificador con id 001 con el que estamos trabajando, tiene como keywords asociadas a su perfil de usuario los valores “negro,comedia,pianista,animal,white”. El número total de keywords para este usuario son 5. El valor del campo es de tipo String, por lo que el valor de cada una de las keywords, está delimitado mediante el uso del carácter coma.

ID_DECODER	ID_USER	USER_KEYS
001	005	negro,comedia,pianista,animal,white

Figura 11 Ejemplo de keywords almacenadas de un usuario

4.4.3 Creación XML petición

Una vez obtenido el valor de las keywords que caracterizan a un usuario, se creará un fichero xml correspondiente a la petición de recomendación, que incluirá las keywords del usuario obtenidas en el anterior punto, junto con las películas disponibles en la parrilla televisiva en ese mismo momento. Para simular el catálogo de películas disponibles, se utilizará otro fichero xml, llamado películasDisponibles.xml, en el cual incluiremos las películas que queramos ofrecer en cada momento.

La estructura y contenido del fichero películasDisponibles.xml será la siguiente:

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <getEncajeSinopsis xmlns="http://ontología.com/encaje">
      <keywordsDestino
id=“14”>noche,opera,comedia,harpo,groucho,risas</keywordsDestino>
      <keywordsDestino
id=“67”>Humphrey,Bogart,paris,pianista,color</keywordsDestino>
      <keywordsDestino
id=“73”>naufrago,isla,tom,cocos,accidente,avion</keywordsDestino>
    </getEncajeSinopsis>
  </soap:Body>
```

</soap:Envelope>

En este fichero, podemos destacar las siguientes etiquetas:

- *<keywordsDestino>: Palabras clave que caracterizan a una película, la cual viene identificada por el atributo "id" dentro de esta etiqueta.*

Las principal etiqueta de este fichero es *<keywordsDestino>*. Donde el valor del atributo *id*, nos proporciona el identificador único de la película a la que hacemos referencia en cada momento, y cuyo valor de etiqueta nos proporciona todas las keys características de la película.

En el momento en el que hemos cargado las keys del usuario de la base de datos, y hemos obtenido las películas disponibles mediante el fichero peliculasDisponibles.xml, crearemos un fichero xml, con el contenido de toda esa información, que nos servirá para enviarla al servidor, proporcionándole así todo lo necesario para que nos proporcione la mejor elección posible.

Este diagrama muestra de manera gráfica como sería el proceso:

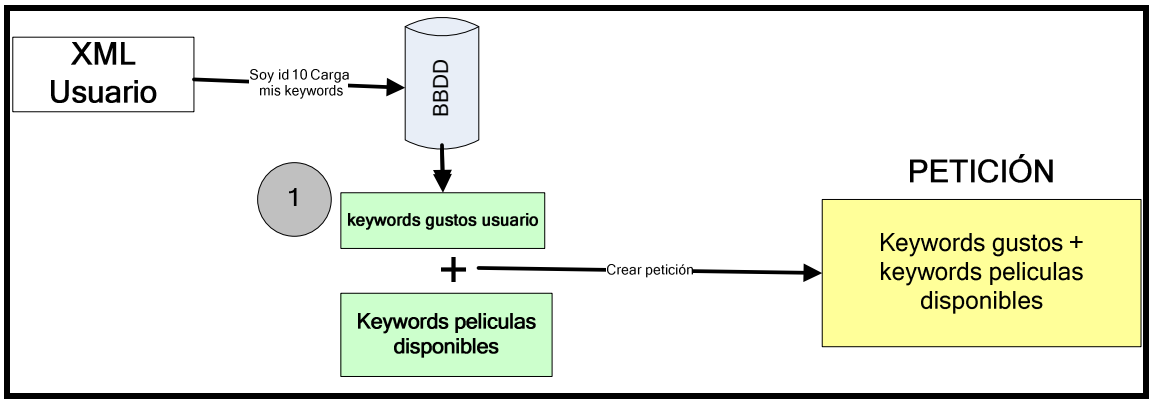


Figura 12 Creación de petición de recomendación

Por simplicidad, el fichero que enviaremos se llamará *peticion.xml*, y tendrá la siguiente estructura:

<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">

<soap:Body>

<getEncajeSinopsis xmlns="http://ontología.com/encaje">

<keywordsFuente id=001>negro,comedia,pianista,animal,white</keywordsFuente>

<keywordsDestino id="14">noche,opera,comedia,harpo,groucho,risas</keywordsDestino>

<keywordsDestino id="67">Humphrey,Bogart,paris,pianista,color</keywordsDestino>

<keywordsDestino id="73">naufrago,isla,tom,cocos,accidente,avion</keywordsDestino>

</etEncajeSinopsis>

</soap:Body>

</soap:Envelope>

Reutilizando la estructura del fichero de películas disponibles, podemos ver como el fichero peticion.xml, tiene la etiqueta *<keywordsDestino>*, donde el valor del atributo *id*, nos proporciona el identificador único de la película a la que hacemos referencia en cada momento, y cuyo valor de etiqueta nos proporciona todas las keys características de la película. Pero además se incluye la etiqueta *<keywordsFuente>* donde el valor del atributo *id* nos proporciona el identificador único del decodificador del usuario para el que estamos realizando la petición, y cuyo valor de etiqueta nos proporciona todas las keys características de éste usuario.

4.4.4 Extracción de keywords

Tras crear el fichero peticion.xml, partiremos de la idea de envío hacia el servidor, donde en el momento de la recepción del mismo, éste extraerá, por un lado, las keywords que definen el perfil del usuario, y por otro las keywords características de cada una de las películas disponibles para la recomendación.

De esta manera crearemos varias instancias de datos mediante una estructura de datos en memoria, donde almacenaremos todo la información referente al usuario y a las películas disponibles.

En total se crearán n+1 listas de keywords, que se corresponderán con las n listas de keywords relacionadas con cada una de las n películas y una lista más, que contendrá las keywords asociadas a los gustos del usuario para el que estamos creando la petición.

4.4.5 Inferencia de sinónimos, hiperónimos e hipónimos

Una vez cargadas todas las keywords, tanto las del usuario, como las de las películas, recorreremos una a una las que componen la lista de keywords de gustos del usuario, buscando en la base de datos los sinónimos, hiperónimos e hipónimos correspondientes a cada una de ellas.

El siguiente dibujo muestra visualmente como obtendremos, a partir de una lista de palabras, una muestra mucho más amplia de keywords recuperadas de la base de datos, que nos permitan definir mas específicamente cuales son los gustos y el perfil del usuario con el que estamos trabajando.

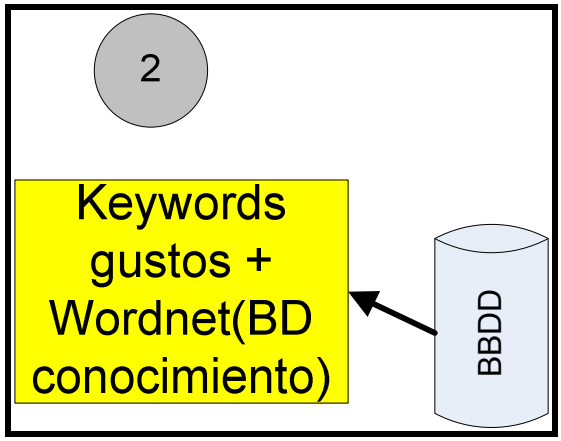


Figura 13 Inferencia de sinónimos, hiperónimos e hipónimos

De esta manera, como se puede observar gráficamente en el siguiente diagrama, para nuestro usuario, anteriormente teníamos cargado una lista con tres palabras, denominadas keyword 1, keyword 2 y keyword 3, y en este estado del programa, se incluirán a cada miembro de la lista de keywords, tres listas más, la de sinónimos, hiperónimos e hipónimos, las cuales estarán compuestas por todos y cada uno de los

sinónimos, hiperónimos e hipónimos que hayamos encontrado para la keyword con la que estamos trabajando en la base de datos.

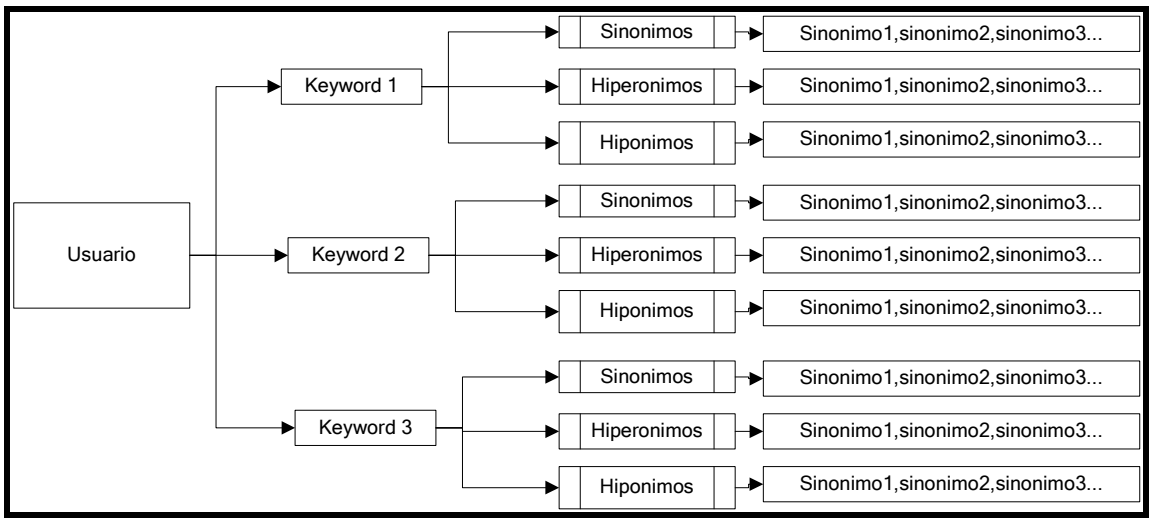


Figura 14 Estructura de almacenamiento de similitudes de keywords en memoria

4.4.6 Asignación de pesos.

En este momento, tenemos por un lado las listas de keywords de las películas disponibles, y por otro lado, una lista "ampliada" con conocimiento inferido a través de la base de datos de sinónimos, de los gustos del usuario para el que estamos trabajando.

Llegado a este punto, pasaremos a evaluar como de afín es una película para los gustos del usuario. Para ello, recorremos una a una las keywords de cada película, e iremos comparando cada una de ellas, con las keywords de los gustos del usuario, asignando de esta manera una puntuación significativa a cada película, dependiendo de la similitud de las keyword de la película con las keywords del usuario.

El modelo de comparación de keywords, podrá llevarse a cabo de dos maneras:

1. La comparación se hará con un simple "equals"
2. La comparación será mediante un algoritmo que implementa la función "like", comparando básicamente el lexema de la palabra o un numero X de letras, dependiendo de la longitud de la palabra.

A continuación se expone un diagrama donde se puede observar de manera gráfica este paso:

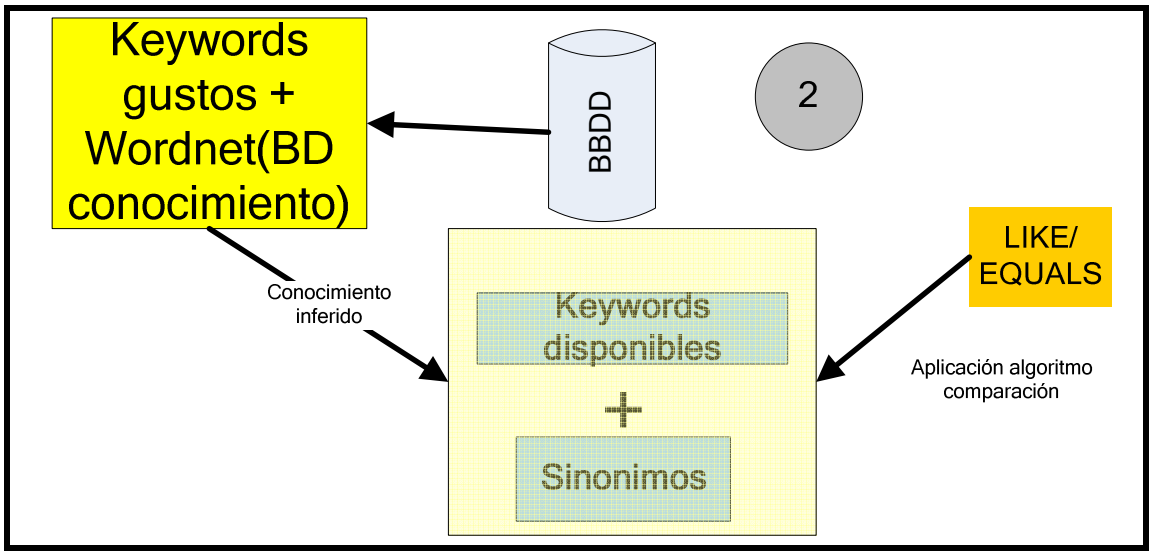


Figura 15 Proceso de asignación de pesos en las keywords

El modelo de puntuación ó asignación de pesos para designar las películas más significativas en el proceso de elección se hará de la siguiente manera:

1. **Evaluación de los sinónimos de la película**: Se recorrerán uno a uno todos las keywords de la película, y se compararán con cada una de los sinónimos del usuario. Para cada sinónimo que coincida con alguna de las keywords de la película, se le asignara la cantidad de 3 puntos a la película que se está evaluando.

2. **Evaluación de los hiperónimos e hipónimos de la película: Se** recorrerán uno a uno todos las keywords de la película, y se compararán con cada una de las hiperónimos e hipónimos del usuario. Para cada hiperónimos e hipónimos que coincida con alguna de las keywords de la película, se le asignara la cantidad de 2 puntos a la película que se está evaluando.

3. **Evaluación de keywords exactas:** Para cada una de las keyword de la película que coincida con alguna keyword del perfil del usuario, se le asignará el peso de 5 puntos a la película que se está evaluando**.**

4.4.7 Elección de la recomendación

Tras asignar a cada película su relevancia correspondiente, se ordenará la lista de películas de manera descendente, atendiendo a la relevancia obtenida, y se elegirán las n primeras para crear la respuesta.

De esta manera, sólo enviaremos las películas que mas relevancia tengan, y por lo tanto las películas que sean mas afines a los gustos del usuario, creando así una recomendación acotada y limitada en número n de películas afines.

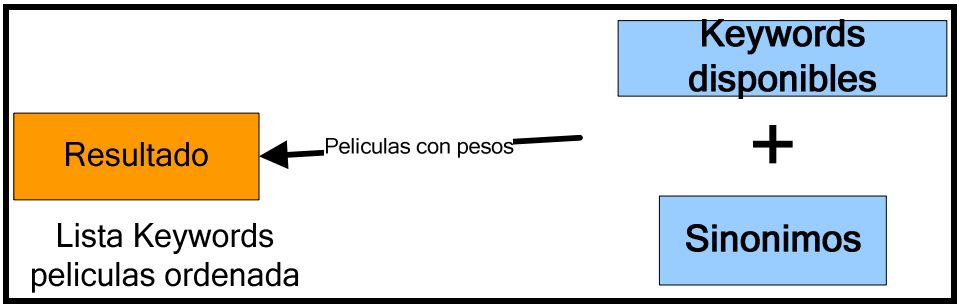


Figura 16 Proceso de ordenación de recomendación

4.4.8 Creación de respuesta

Para la creación de la respuesta de recomendación, se generará un fichero xml, con las películas recomendadas, atendiendo a la petición que nos llego al inicio de la ejecución.

Suponiendo que tenemos dos películas para recomendar, denotadas como “C” y “B”, A continuación se muestra un ejemplo del contenido y estructura del fichero xml que se generará para la respuesta de recomendación:

3: XML SOAP → Los dos que mejor encajan son C y B

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
 <soap:Body>
  <setPreferencias xmlns="http://ontología.com/encaje">
   <recomendado orden=1>C</recomendado>
```

```
    <recomendado orden=2>B</recomendado>
  </setPreferencias>
 </soap:Body>
</soap:Envelope>
```

Podemos observar en el ejemplo de este archivo xml, que las películas vienen ordenadas por orden de preferencia, según el peso obtenido en la evaluación de éstas con respecto a las características del usuario, obteniendo en primer lugar la película C y en segundo lugar la película B. Interpretando de esta manera que la película C será más interesante para el usuario que la B.

4.4.9 Envío de la recomendación

Tras crear la respuesta en la parte del servidor, se generará la recomendación al usuario, la cual contendrá los títulos y la información relevante de las películas recomendadas. Para ello, se realizarán consultas a la base de datos donde, a partir de cada id de las películas obtenido en la respuesta, se obtendrán todos los datos relativos a éstas para presentarlos al usuario final.

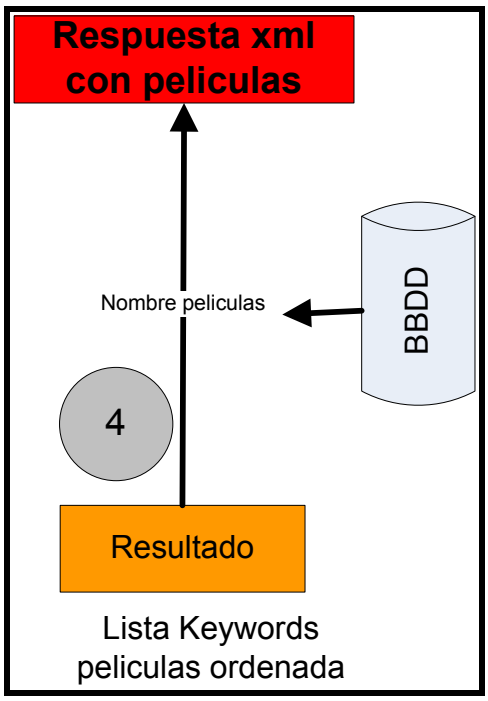


Figura 17 Proceso de envío de la recomendación

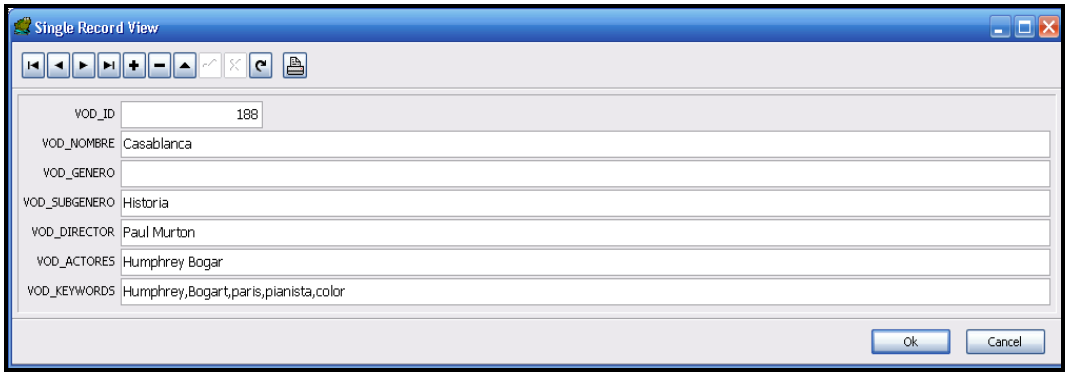


Figura 18 Ejemplo de almacenamiento datos de películas

La recomendación se enviará al usuario final, mediante un fichero xml, que tendrá el siguiente formato:

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
 <soap:Body>
  <setNoTePierdas xmlns="http://modelousuario.com/recomendados">
   <NoTePierdas orden="1">Casablanca (tve1)</NoTePierdas>
   <NoTePierdas orden="2">Los hermanos Marx (tve2)</NoTePierdas>
  </setNoTePierdas>
 </soap:Body>
</soap:Envelope>
```

4.5 Diseño Base de Datos

Para inferir todo el conocimiento relativo a sinónimos, hipónimos e hiperónimos, nos apoyaremos en Wordnet, una base de datos donde se almacena toda esta información.

En un principio se trabajó con una instancia de Wordnet bajo Access, pero tras comprobar que los tiempos de respuesta eran bajos, se decidió migrar la base de datos de Access a Oracle.

Todos las palabras se encuentran en la tabla SENTIDO, donde se indica el tipo de palabra, y su código(SYNSET_ID), y es a partir de este código, donde relacionamos en las demás tablas de la base de datos las relaciones de sinonimia, hiperonima o hiponima que mantienen entre sí.

A continuación se muestra un diseño de la base de datos de conocimiento de Wordnet implementado bajo Oracle.:

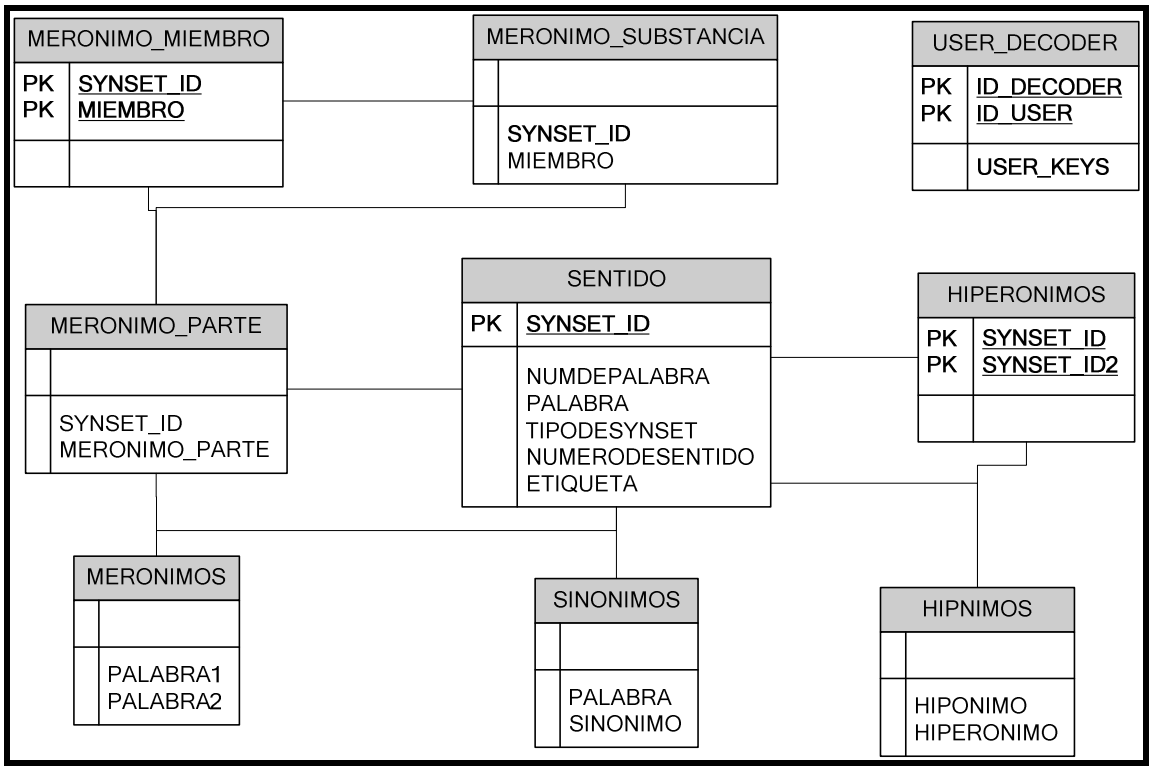


Figura 19 Diagrama de base de datos de Wordnet

Tabla Meronimo_XXX: tablas relativas a palabras y sus meronimos correspondientes.

Tabla Hiperonimos: Tabla que contiene dos synset, que se relacionan mutuamente mediante la relación de hiperonimia.

Tabla sentido: Significado del synset.

Tabla user_decoder: Contiene el identificador de un cliente, junto con su decodificador, y las keywords asociadas a dicho cliente.

4.6 Diseño E/R

A continuación se muestra un diagrama E/R sobre el diseño de la base de datos a partir de la cual vamos a inferir conocimiento de sinónimo, meronimos, hipónimos e hiperónimos de palabras:

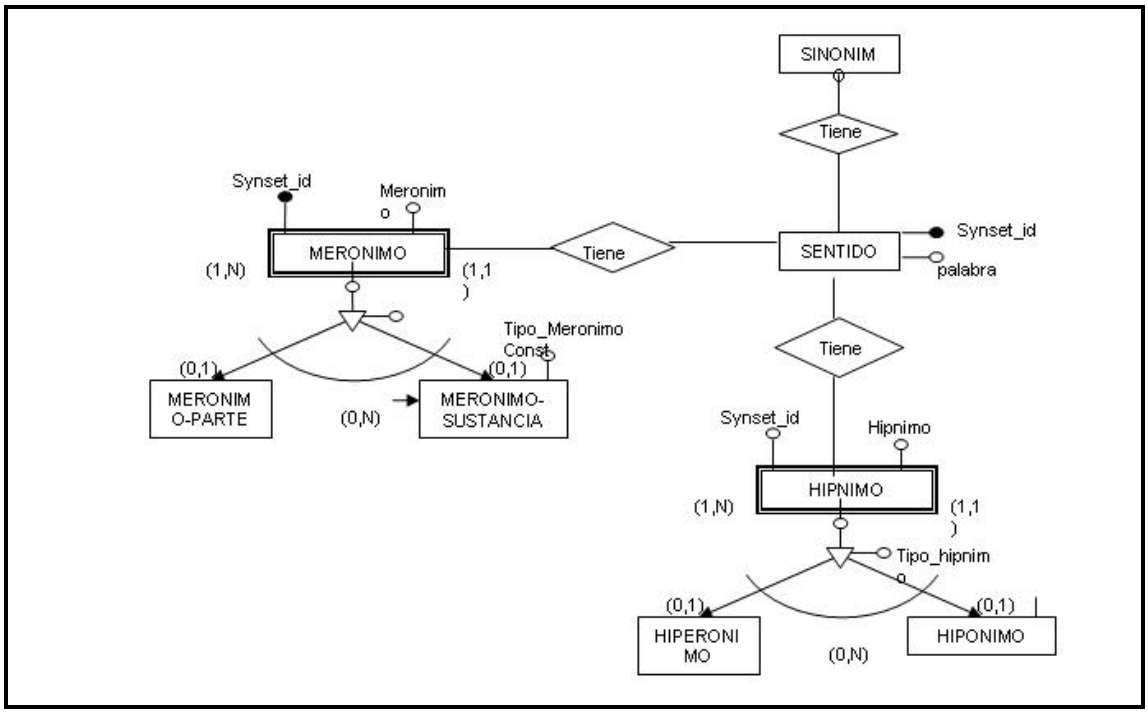


Figura 20 Diagrama E/R

4.7 Diseño consultas

En este apartado vamos a explicar todas las consultas que se han diseñado para poder inferir todo el conocimiento de la aplicación sobre un usuario.

- Inferir conocimiento a partir de sinónimos de keywords:

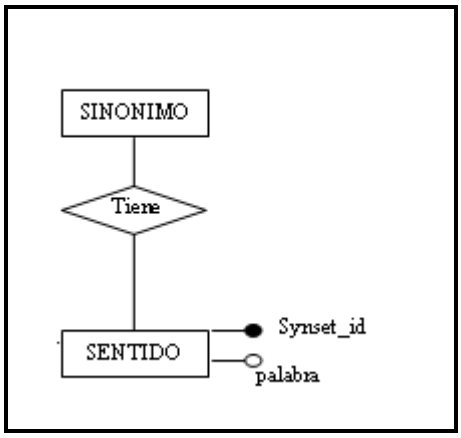
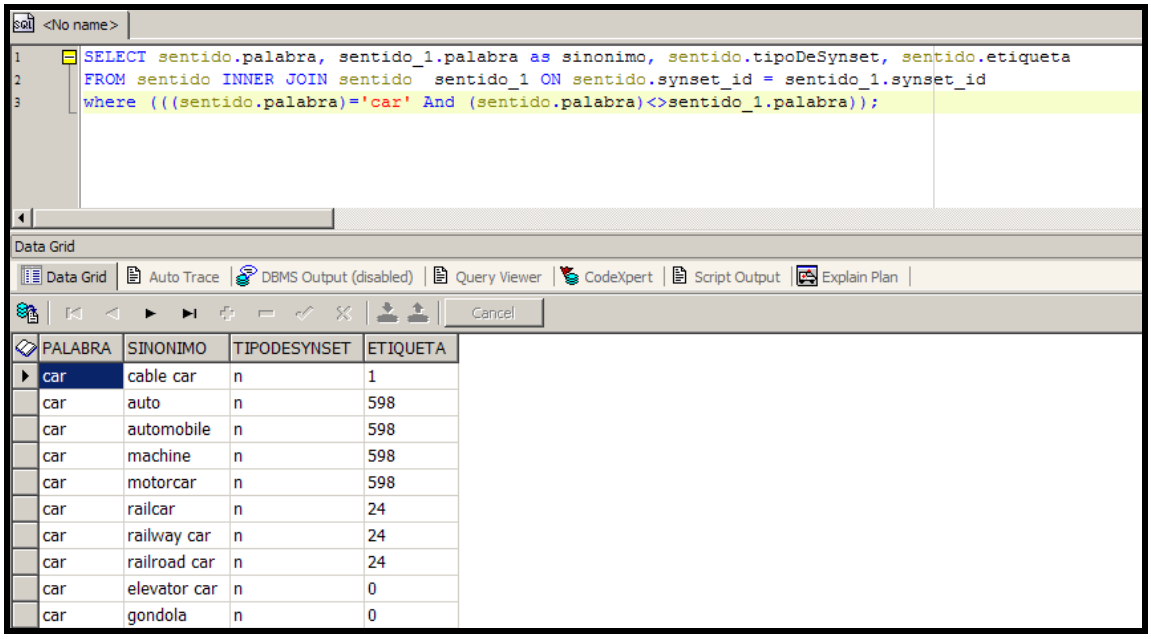


Figura 21 E/R sinónimo-sentido.

```
SELECT sentido.palabra, sentido_1.palabra as sinonimo, sentido.tipoDeSynset, sentido.etiqueta
FROM sentido INNER JOIN sentido  sentido_1 ON sentido.synset_id = sentido_1.synset_id
where (((sentido.palabra)='car' And
(sentido.palabra)<>sentido_1.palabra));
```



PALABRA	SINONIMO	TIPODESYNSET	ETIQUETA
car	cable car	n	1
car	auto	n	598
car	automobile	n	598
car	machine	n	598
car	motorcar	n	598
car	railcar	n	24
car	railway car	n	24
car	railroad car	n	24
car	elevator car	n	0
car	gondola	n	0

Figura 22 Consulta sinónimos.

El tiempo de ejecución de la anterior consulta, es de aproximadamente 1 segundo. Teniendo en cuenta que solo lo estamos ejecutando para una keywords, el tiempo de ejecución para un número mayor de keywords sería muy grande.

Si creamos una vista materializada a partir de la anterior consulta, la aplicación sólo tendría que ejecutar la siguiente consulta:

```
SELECT sinonimo  from SINONIMOS where palabra = 'car'
```

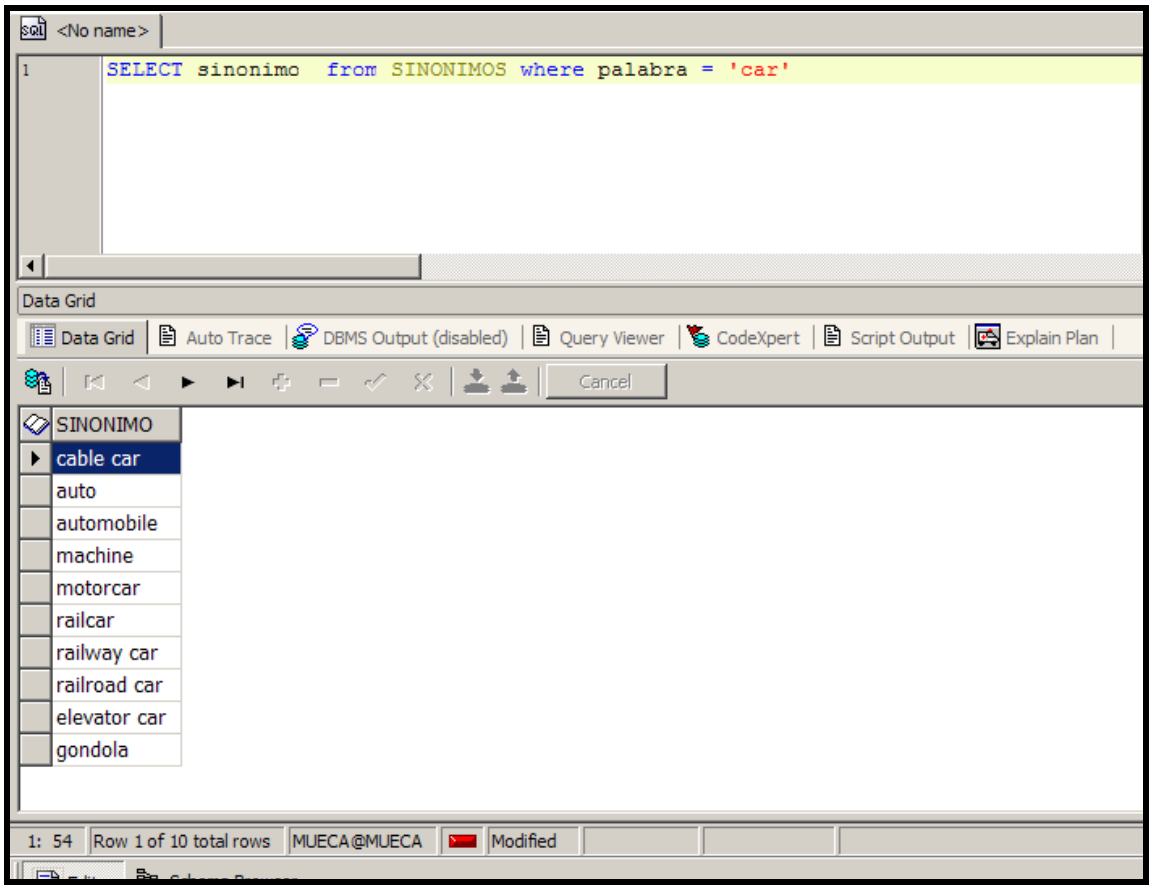


Figura 23 Consulta sinónimos(Vista materializada).

El tiempo de ejecución de la anterior consulta, es de milésimas de segundo, un tiempo más que optimizado con respecto al tiempo de ejecución de la consulta sobre varias tablas para inferir los sinónimos de una keyword de usuario.

- Inferir conocimiento a partir de merónimos keywords:

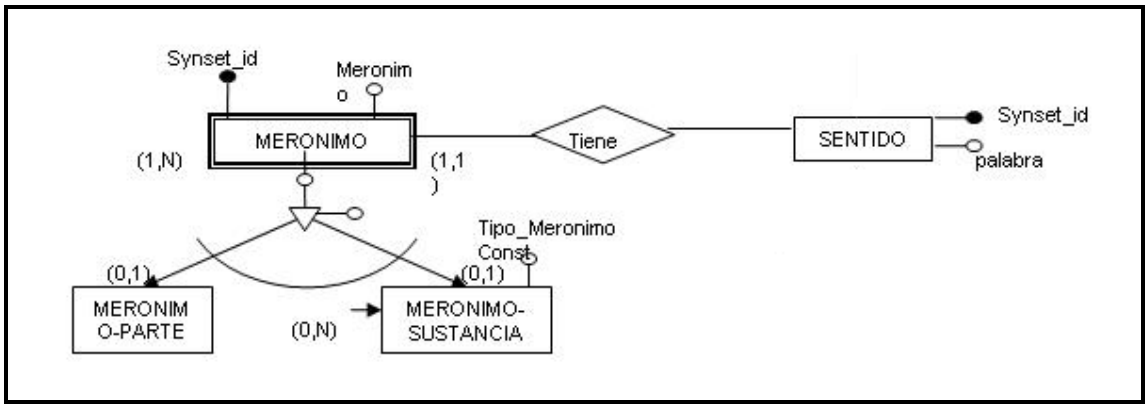


Figura 24 Diagrama E/R meronimo.

```
SELECT sentido.palabra palabra1, sentido_1.palabra palabra2
FROM sentido sentido_1 INNER JOIN (sentido INNER JOIN meronimo_parte
ON sentido.synset_id = meronimo_parte.synset_id)
ON sentido_1.synset_id = meronimo_parte.meronimo_parte
WHERE (((sentido.palabra)='car')) OR
(((sentido_1.palabra)='car'));
```

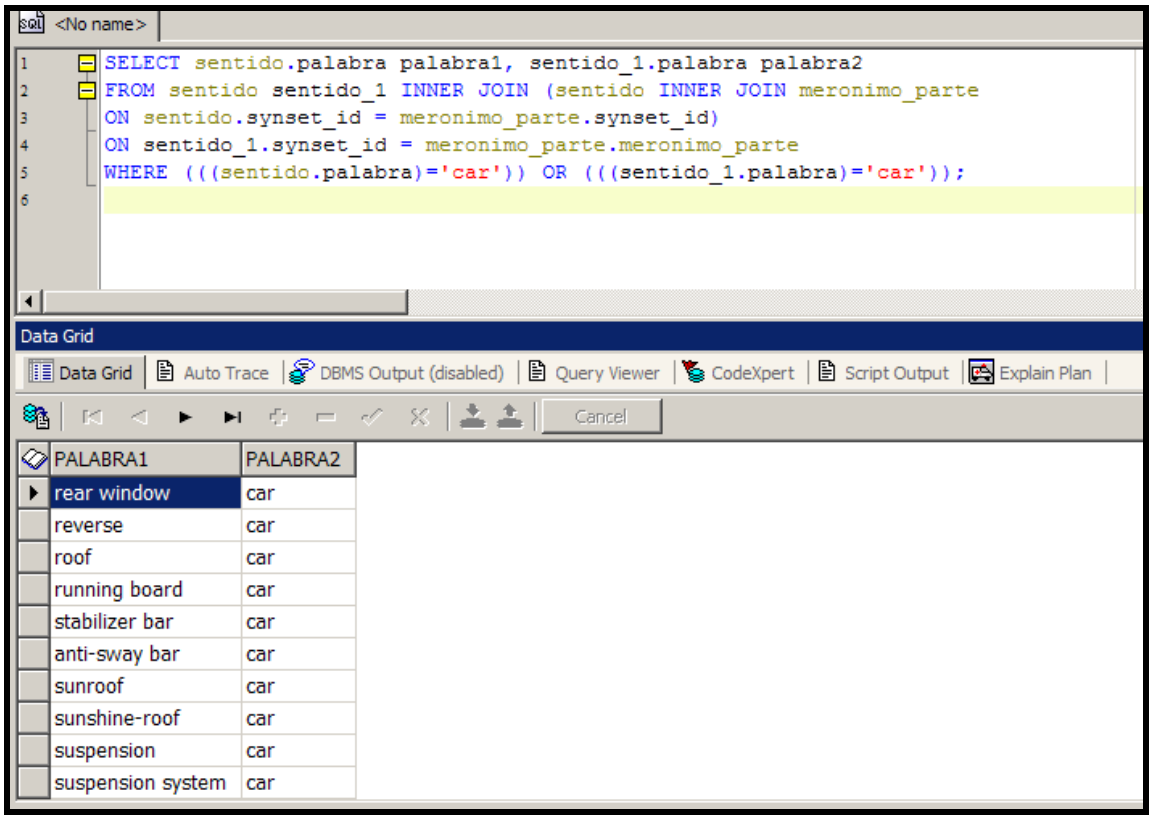


Figura 25 Consulta merónimos.

El tiempo de ejecución de la anterior consulta, es de aproximadamente 200 milisegundos. Teniendo en cuenta que solo lo estamos ejecutando para una keywords, el tiempo de ejecución para un número mayor de keywords sería muy grande.

Si creamos una vista materializada a partir de la anterior consulta, la aplicación sólo tendría que ejecutar la siguiente consulta:

```
SELECT palabra1,palabra2  from MERONIMOS where palabra1 = 'car' OR palabra2 = 'car'
```

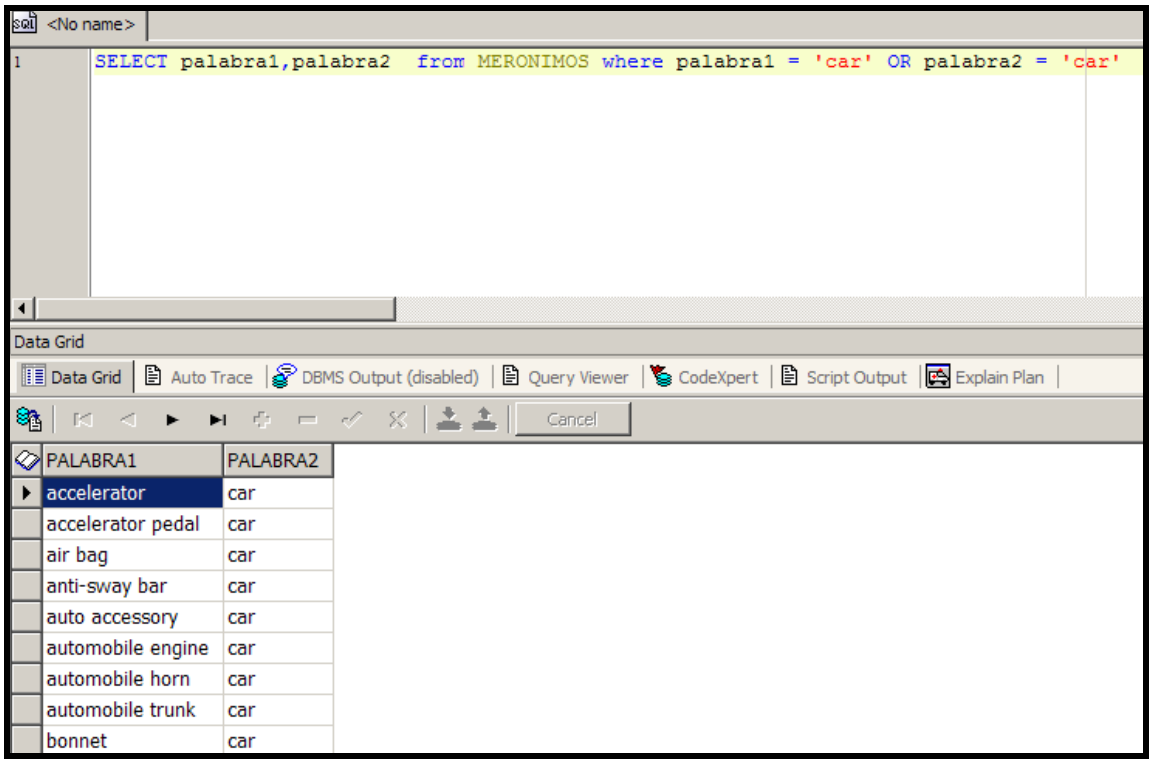


Figura 26 Consulta Merónimos(Vista Materializada).

El tiempo de ejecución ahora, es de 65 milisegundos, con lo que el tiempo se ha optimizado de manera notable.

- Inferir conocimiento a partir de hiperónimos e hipónimos de keywords:

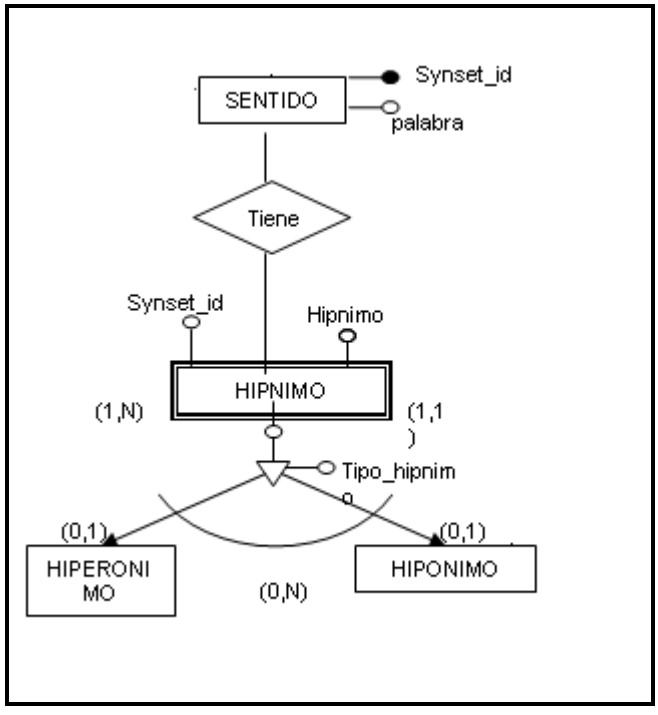


Figura 27 Diagrama E/R Hipnimo.

```
SELECT sentido.palabra HIPONIMO, sentido_1.palabra HIPERONIMO
FROM sentido sentido_1 INNER JOIN (sentido INNER JOIN hiperonimos
ON sentido.synset_id = hiperonimos.synset_id1) ON sentido_1.synset_id
= hiperonimos.synset_id2
WHERE (((sentido.palabra)='car')) OR (((sentido_1.palabra)='car'));
```

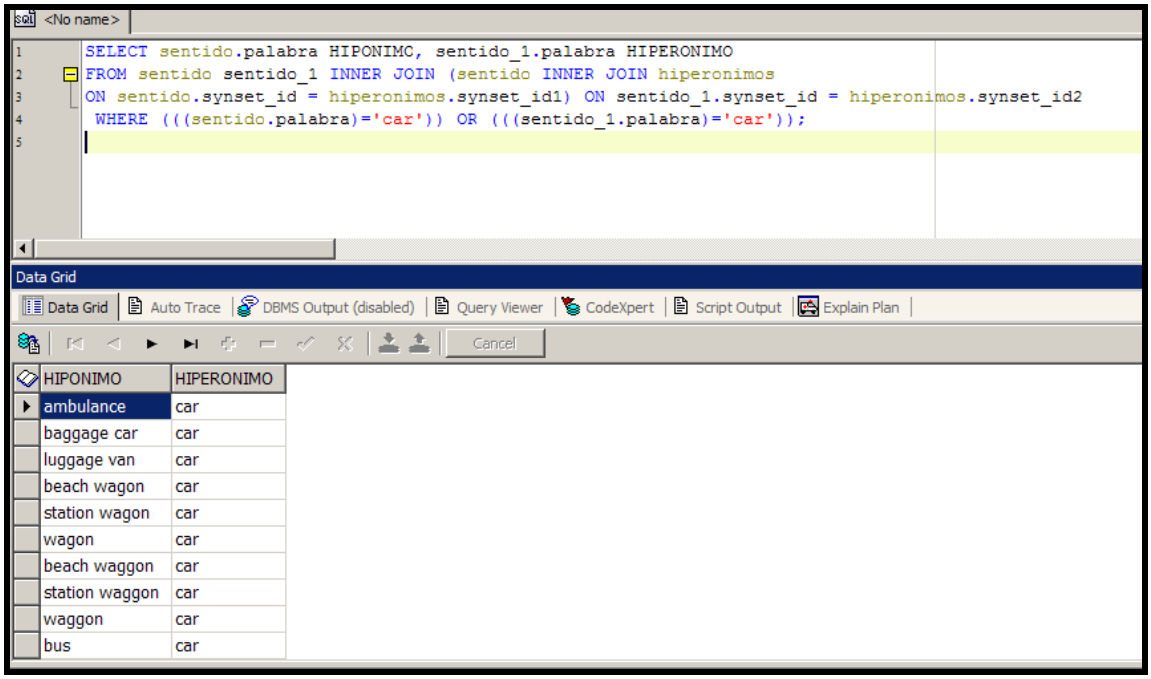


Figura 28 Consulta Hipnimos.

El tiempo de ejecución es de 1 segundo.

Si creamos una vista materializada a partir de la anterior consulta, la aplicación sólo tendría que ejecutar la siguiente consulta:

```
SELECT hiponimo, hiperonimo  from HIPNIMOS where hiponimo = 'car' or hiperonimo = 'car'
```

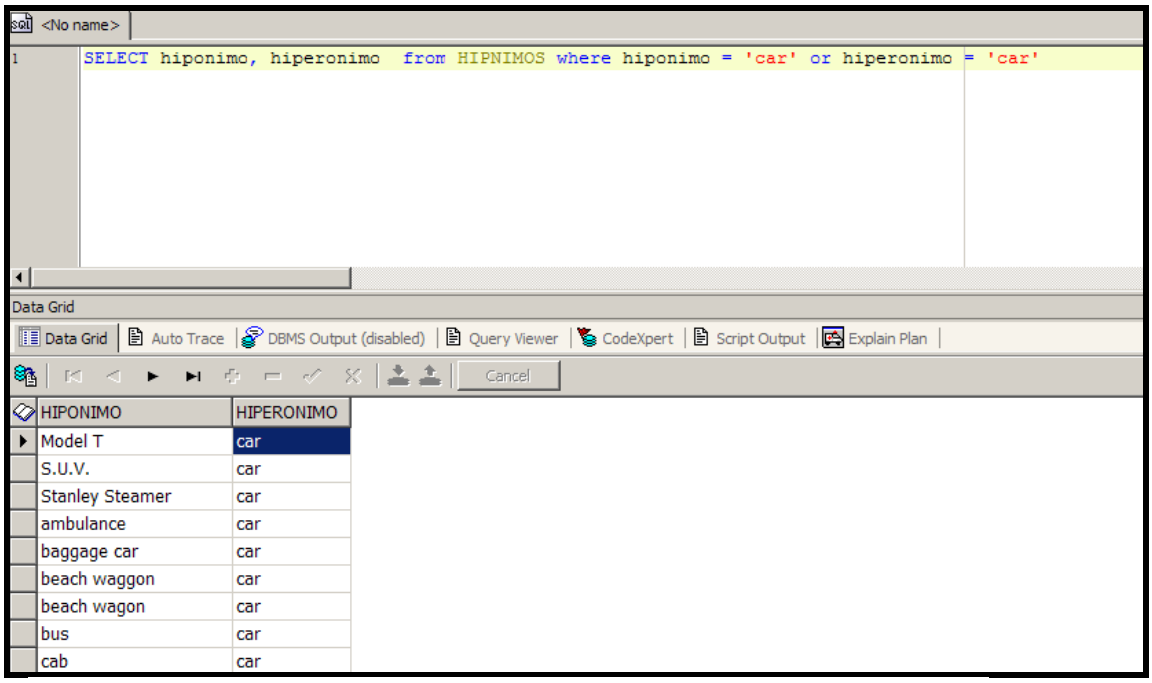


Figura 29 Consulta Hipnimos(Vista Materializada)

El tiempo de ejecución ahora, es de 406 milisegundos, con lo que el tiempo se ha optimizado de manera notable.

Por otra parte, tenemos otra base de datos, donde almacenamos los datos de cada película (actores, directores, titulo….) la cual nos servirá para, a partir del id de las películas que se van a recomendar en la respuesta, obtener los títulos de éstas, para crear la recomendación con la información necesaria, y enviarla al usuario.

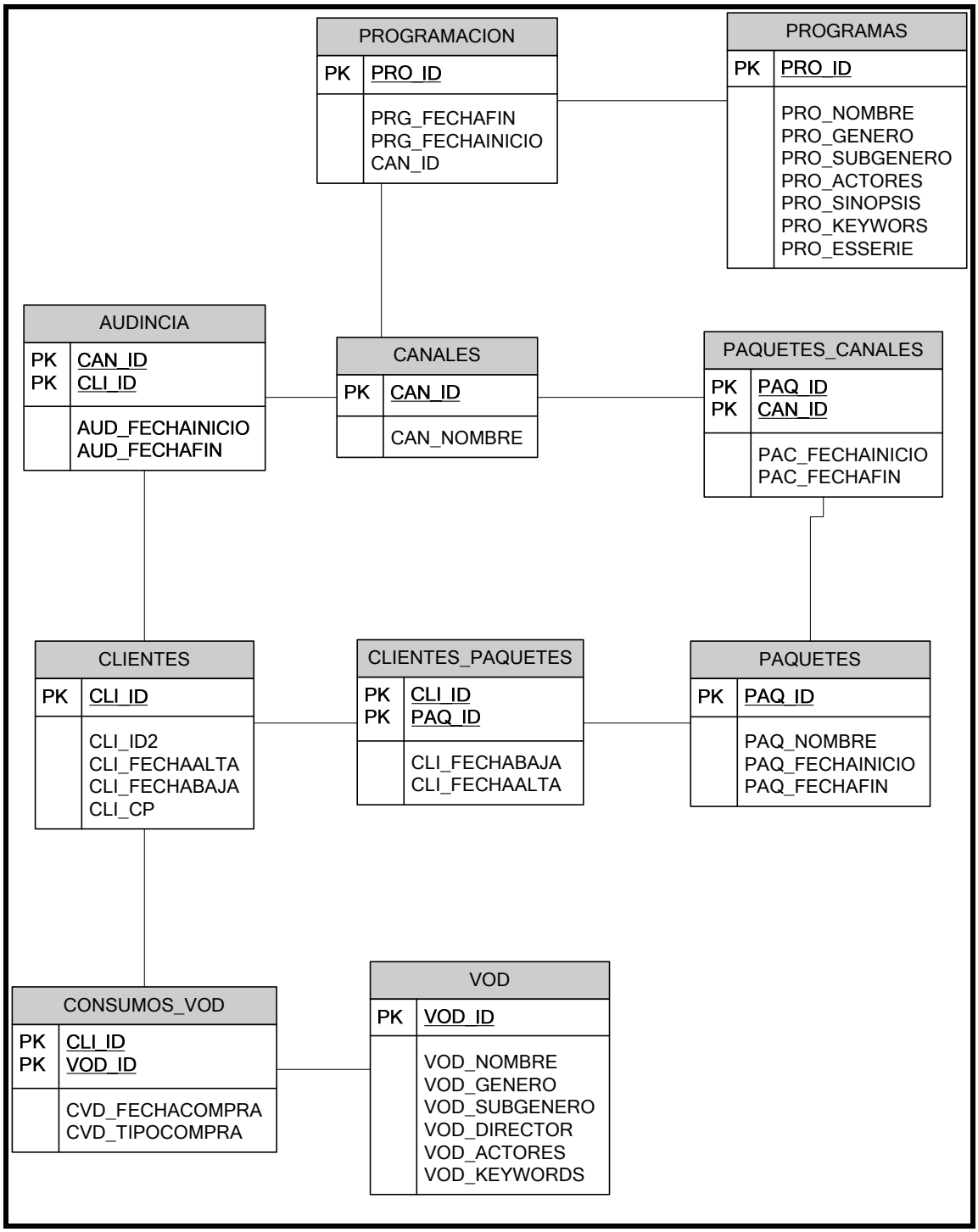


Figura 30 Diagrama de base de datos de películas

Capítulo 5 - Estudio de la eficacia y la eficiencia del sistema

5.1 Eficacia de la aplicación

Para comprobar la eficacia del sistema, se han realizado numerosas pruebas, cuyos datos y estadísticas se presentan más adelante.

5.2 Eficiencia de la aplicación

Para medir la eficiencia de la aplicación se ha determinado que la magnitud más importante es el tiempo de inferencia de sinónimos sobre las keywords de usuario.

5.2.1 Marco de las pruebas de eficiencia

Para estudiar el tiempo consumido por el sistema en el proceso de respuesta ante una petición del usuario, se han realizado numerosas mediciones de los tiempos de respuesta sobre los diferentes procesos en los que se compone el sistema, en función del tamaño de las keywords del usuario, y el número de películas disponibles.

Tras realizar numerosas pruebas contra la base de datos de Wordnet, implementada en Access, se comprobó que los tiempos de respuestas podrían ser minimizados considerablemente si migraba a Oracle, por lo que finalmente se realizó dicha migración.

Aún habiendo migrado la base de datos a Oracle, se verificó que los tiempos de respuesta no eran óptimos, por lo que se estudiaron varias acciones, dentro de las cuales se crearon índices en las tablas, y una serie de vistas materializadas con los principales campos a consultar, lo cual redujo de manera muy considerable el tiempo de respuesta de la aplicación, consiguiendo así unos resultados óptimos para el usuario.

5.2.2 Pruebas de eficiencia realizadas y datos obtenidos

De esta manera, se han realizado numerosas baterías de pruebas para estudiar y comparar los tiempos de inferencia de sinónimos de las palabras clave de un usuario, así como el tiempo de respuesta final para el usuario.

A continuación vamos a constatar la eficiencia del sistema, mostrando los resultados obtenidos en varias fases del desarrollo del proyecto, pudiendo comparar así las optimizaciones en tiempo y recursos del sistema obtenidos.

Vamos a dividir las pruebas principalmente en 2 fases, siendo la primera cuando el estado de la aplicación no contaba con la migración de la base de datos de Wordnet a Oracle, ni con la creación de los índices en tablas y las vistas materializadas, y la segunda fase contemplando estas mejoras en el proyecto.

Para comprobar la eficiencia del sistema, vamos a dividir las pruebas, dependiendo del número de keywords y películas disponibles, aumentando de 5 en 5 el número de éstas.

FASE 1

En esta fase se estudian los tiempos de respuesta de la aplicación, trabajando con Wordnet bajo Access, y sin realizar mejoras de ningún tipo, sobre índices y vistas materializadas en la base de datos de Wordnet bajo Oracle.

Se han realizado numerosas baterías de pruebas, estudiando en cada una de ellas el impacto del tiempo de respuesta en el número de keywords del usuario en el número de películas disponibles en la parrilla televisiva.

1. Esta batería va a mostrar el impacto que tienen el aumento del número de keywords de un usuario, frente al número constante de películas disponibles en la parrilla televisiva.

   Para plasmar el tiempo de ejecución del proceso final de recomendación, así como cada uno de los tiempos de las fases que conforman el proceso final, se va a utilizar una tabla formada por varias columnas donde se muestra:

   - Nº keywords: Numero total de keywords que existen en la base de datos relacionadas con el usuario que se esta trabajando
   - Nº películas: Número de películas disponibles en la parrilla televisiva en el momento de realizar la recomendación.
   - P1: Tiempo de ejecución para realizar la primera fase, consistente en la carga de las keywords de un usuario de la base de datos, a

partir de id del decodificador para el que se esta realizando la recomendación.

- P2: Tiempo de ejecución para la realizar la segunda fase, consistente en la carga de sinónimos para las keywords del usuario, y la inferencia de éstas sobre Wordnet.
- P3: Tiempo de ejecución para realizar la tercera fase, en la cual se comparan las keywords del usuario con las keywords de las películas, asignando pesos a las películas mas afines.
- P4: Tiempo de ejecución para realizar la cuarta fase, en la cual se crea un archivo xml con la respuesta de la recomendación para el usuario final, donde se incluyen las recomendadas.
- TOTAL: Tiempo total de ejecución de todo el proceso completo.

Todos los tiempos que se muestran en la siguiente tabla, están medidos en segundos.

Nº keywords	Nº películas	P1 (s)	P2 (s)	P3 (s)	P4 (s)	T.TOTAL (s)
5	5	0,3	24	0,01	0,02	24,33
10	5	0,3	45	0,01	0,02	45,33

Tabla 41 - Tiempos de carga aumentando el num de keywords..

Observando los tiempos de ejecución, podemos constatar que los tiempos de ejecución para 5 y 10 keywords de usuario son inadmisibles para llevar a cabo el proceso, ya que no se puede tener a un usuario este tiempo esperando a una recomendación televisiva, ya que probablemente se cansaría de esperar y cancelaría la ejecución. El problema de retardo estaba en el proceso 4, correspondiente a la inferencia de sinónimos de las keywords del usuario, proceso que interaccionaba constantemente con la base de datos, por lo que se tomo la decisión de optimizar las consultas a la base de datos para poder reducir el tiempo de ejecución lo máximo posible.

FASE 2

En esta fase se estudian los tiempos de respuesta de la aplicación, trabajando con Wordnet migrado a Oracle y con las optimizaciones de índices y vistas materializadas sobre la base de datos de sinónimos en Wordnet.

Se han realizado numerosas baterías de pruebas, estudiando en cada una de ellas el impacto del tiempo de respuesta en el número de keywords del usuario en el número de películas disponibles en la parrilla televisiva.

1. Esta batería va a mostrar el impacto que tienen el aumento del número de keywords de un usuario, frente al número constante de películas disponibles en la parrilla televisiva.

Nº keywords	Nº películas	P1 (s)	P2 (s)	P3 (s)	P4 (s)	T.TOTAL (s)
5	5	0,015	1,43	0,01	0,01	1,45
10	5	0,1	1,46	0,02	0,01	1,59
15	5	0,01	2,1	0,01	0,01	2,12
20	5	0,015	2,2	0,01	0,01	2,23

Tabla 42 - Tiempos de carga aumentando el num de keywords..

Si observamos los tiempos obtenidos tras la optimización de las consultas a la base de datos, podemos concluir que se ha reducido de forma considerable el tiempo de ejecución del proceso 2, proceso que producía el retardo en la recomendación en la anterior fase cuando no se había optimizado nada. Si comparamos estos resultados con los anteriores, tenemos un resultado de 1,45 frente a 24,33 segundos, con lo cual, se ha reducido en casi 17 veces, considerando así un tiempo de ejecución aceptable para la espera de un usuario ante el resultado de una recomendación.

El siguiente diagrama muestra el aumento del tiempo de ejecución total, con respecto al aumento del número de keywords del usuario, teniendo un número constante de películas disponibles en la base de datos:

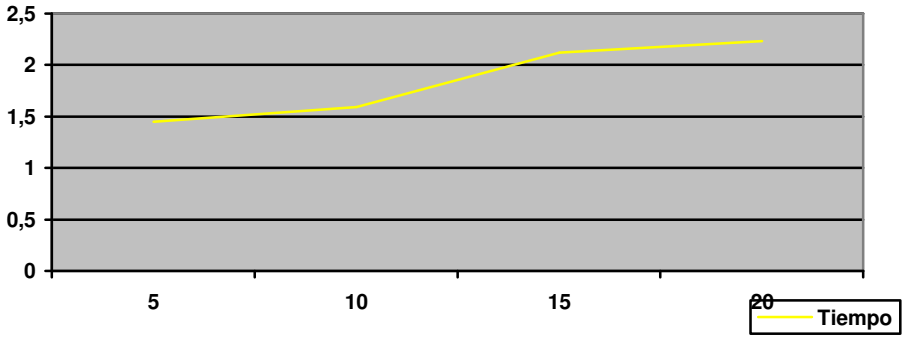


Tabla 43 - Tiempos totales de carga

En el diagrama anterior (Tabla 43) podemos observar como el tiempo de ejecución total el proceso de recomendación va aumentando conforme aumenta el número de keywords disponibles del usuario.

2. La siguiente batería de pruebas, va mostrar el pacto que tiene el aumento del número de películas disponibles en la parrilla televisiva, frente al número constante de keywords de un determinado usuario.

Nº keywords	Nº películas	P1 (s)	P2 (s)	P3 (s)	P4 (s)	T.TOTAL (s)
10	5	0,01	1,4	0,01	0,02	1,35
10	10	0,01	1,4	0,01	0,02	1,35
10	15	0,01	1,4	0,01	0,02	1,35
10	20	0,01	1,4	0,01	0,02	1,35

Tabla 44 - Tiempos de carga aumentando el num de keywords..

Observando los tiempos de ejecución de la tabla anterior (Tabla 44), podemos decir que no es prácticamente apreciable en el tiempo de ejecución el aumento del número de películas frente al número constante del número de keywords del usuario, no siendo así lo que sucedía en el punto anterior con el aumento del número de keywords del usuario.

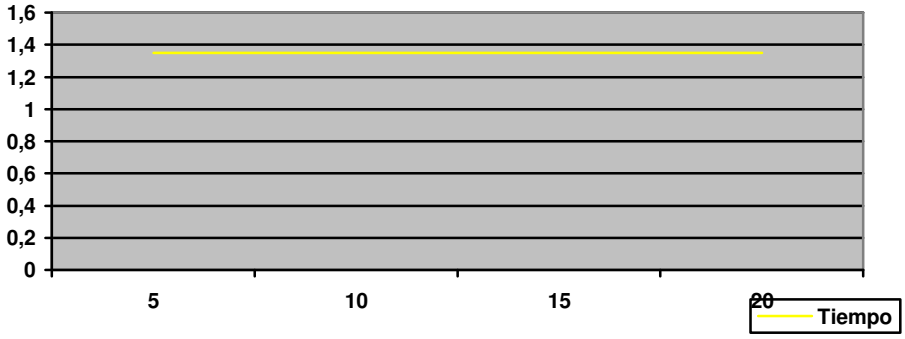


Tabla 45 - Tiempos totales de carga.

En el diagrama anterior (Tabla 45) podemos observar como el tiempo de ejecución total el proceso de recomendación es constante frente al aumento del número de películas disponibles en la base de datos.

Capítulo 6 - Conclusiones y líneas futuras

En este proyecto fin de carrera se ha realizado un recomendador televisivo, basado en el modelo de usuario. Podemos darnos cuenta del gran abanico de programas y utilidades que actualmente existen en el mercado relacionadas con la gestión e interacción televisiva con el usuario, como por ejemplo:

- Proyecto iMEDIA[8].
- Mayordomo de InOutTV[9].
- MythTV[10].
- SmartNavigator de Predictive Networks[11]..
- TV FINDER12:
- AVATAR::

El principal objetivo de este trabajo consistía en desarrollar una aplicación que facilitase el manejo de los distintos contenidos que ofrece la televisión digital de manera que el usuario se sintiese cómodo a la hora de usarla. Podemos decir que este objetivo se ha logrado con la realización de este proyecto fin de carrera. El sistema construido va aprendiendo y retroalimentando los gustos del usuario con el que interactúa pudiendo sugerir día a día las recomendaciones para cada jornada.

La aplicación, apoyada en Wordnet, recoge multitud de términos similares (homónimos, hiperónimos, …) a las keywords originales que proporciona el usuario, infiriendo así nuevas cualidades y reuniendo de esta manera un conjunto mayor de características que ayudan a conocer los gustos del usuario.

De esta manera, podemos destacar una serie de puntos que hacen destacar a este proyecto:

---

[8] [http://imedia.intranet.gr/].

[9] [http://www.inout.tv].

[10] [http://www.mythtv.org]

[11] [http:// www.predictivenetworks.com/]

[12] [ttp://www.tmtfactory.com/articulos/articuloTVFinder200310.pdf]

- Gran abanico de programas y utilidades que actualmente existen en el mercado relacionadas con la gestión e interacción televisiva con el usuario.
- Al estar vinculado en el propio descodificador de una tv de pago (Ono, Canal +….) se puede promocionar como un paquete de servicios.
- Rapidez y fácil acceso a la hora de conectarse al televisor.
- Originalidad, a la hora de ofrecer una "televisión a la carta", donde sabemos que dentro de la misma están todos los programas que nos gustan.
- El usuario tiene un programa que le ayuda a elegir sus programas en función a sus preferencias.
- Configuración inicial de los perfiles. En una casa puede haber cuatro miembros de una familia por lo que se configuraría para cada uno (edad, aficiones, gustos). De aquí podemos destacar el Autocontrol que los padres pueden ejercer al configurar el programa para sus hijos.

Aunque el presente proyecto se presenta como un prototipo de recomendación televisiva eficaz y eficiente, algunos trabajos futuros que aumentarían la robustez, y podrían crear recomendación mas personalizadas, proporcionándole nuevas funcionalidades, podrían ser las siguientes:

- Añadir contenido semántico para las keywords, mediante ontologías.
- Autoalimentación de ontologías para el usuario, aumento de conocimiento actualizado de los gustos de un usuario.
- Descripción de palabra o lista de sinónimos asociado a un sinónimo.
- Envío real de la petición inicial del usuario a un servidor real.
- Utilización de un password para que solo pueda acceder el usuario del que está cargado el perfil.
- Creación de un mando especifico con los botones propios del sistema y con los específicos para cada usuario.

- Al estar filtrado por programas favoritos de cada usuario se podría utilizar para poder sacar los datos de audiencia, por lo que ayudaría a saber los gustos de cada usuario en función de su área geográfica, edad, sexo y preferencias.
- En caso de que los miembros de una misma familia se junten para ver la televisión ¿que programación se ofrece? El programa recoge las palabras comunes de todos los miembros de la familia para así poder recoger los programas afines a todos.
- Obtención de sinónimos para las keywords de las películas además de las keywords del usuario.
- Por último, a pesar de que en varias partes de este documento se habla de modelo de usuario, el sistema desarrollado únicamente utiliza perfiles guiándose de los gustos televisivos, pero en ningún momento integra un modelo como tal. Sería deseable integrarlo para conseguir una funcionalidad completa.

Bibliografía

- [Bourret, R., 2000]. XML and Databases. November, 2000. Accesible en http://www.rpbourret.com/xml/XMLAndDatabases.htm. 4-4-2001.

- [Dejesus, E., 2000]. XML Enters the DBMS Arena. Computerworld. October, 2000. Accesible en http://www.computerworld.com/cwi/story/0,1199,NAV47_STO53026,00.html. 4-4- 2001.

- [Urman, S., 2002]. Oracle9i PL-SQL programming. McGraw-Hill

- [W3C, 2000]. Extensible Markup Language (XML) 1.0 (Second Edition). W3C

- [Resinas M, Fernández P., 2005]. Accesible enhttp://www.tdg-seville.info/pupils/presentations/IntroduccionWebSemantica.pdf

- [Vidal, E., 2008]. Integración de datos en el Internet.. Accesible en http://www.lsi.upc.es/~www-si/seminari/abstrSeminari/MEVidal.pdf

- [Bonnici, S., 2003]. Which Channel Is That On? A Design Model for Electronic Programme Guides. Actas de la European Conference on Interactive Television: from Viewers to Actors? Brighton, UK.

- [Chorianopoulos, K., Lekakos, G. y Spinellis, D., 2003]. The virtual channel model for personalized television. Actas de la European Conference on Interactive Television: from Viewers to Actors? Brighton, UK.

- [Gauntlett, D. y Hill, A., 1999]. TV Living. London, UK: Routledge.

- [Marshall, L., 1997]. The World According to Eco. Wired 5 (3).

- [Pérez de Silva, J., 2002]. La televisión ha muerto. La nueva producción audiovisual en la era de Internet: La tercera revolución industrial. Barcelona, España: Gedisa.

- [Rose, F., 2003]. The Fast-Forward, On-Demand, Network-Smashing Future of Television. Wired 11 (10), 158-163.

- [Schatzberger, R., 2002]. Channel Flipping: The EPG as a Way of Life. UsableiTV 3, 14-16.

- [Serco Usability Services, 2000], Interactive TV and Electronic Programme Guides: Usability Guidelines. Disponible en [http://www.usability.serco.com].

- [Blanco Fernández Y., Pazos Arias J. J., Gil Solla A., Ramos Cabrer M., Barragáns Martínez B. and López Nores M., 2004]. A Multi-Agent Open Architecture for a TV Recommender System: A Case Study using a Bayesian Strategy. In Proc. of the IEEE Sixth International Symposium on Multimedia Software Engineering, 2004.

- [Blanco Fernández Y., Pazos Arias J. J., Gil Solla A., Ramos Cabrer M., López Nores M. and Barragáns Martínez B., 2005]. AVATAR: Modeling Users by Dynamic Ontologies in a TV Recommender System based on Semantic Reasoning. In Proc. of the 3rd European Conference on Interactive Television: User Centred 1W Systems, Pro grammes and Applications (EuroIW-05).

- [Daconta M., Obrst L. J. and Smith K. J. 2003]. The Semantic Web: A Guide to the Future of XML, Web Services and Knowledge Management. John Wiley & Sons.

- [ETSI TS 101 812., 2002] Multimedia Home Platform (MHP).

- [Geroimenko V. and Chen C., 2003]. Visualizing the Semantic Web. Springer Verlag.

- [McGuinness D. and van Harmelen E., 2004] OWL Web Ontology Language Overview. W3C Recommendation.

- [Pagani M., 2003] Multimedia and Interactive Digital TV. Managing the Opportunities Created by Digital Convergence. Idea Group Publishing.

- [Ricci F., Arslan B., Mirzadeh N. and Venturini A., 2002]. ITR:a Case-Based Travel Advisory System. In Proc. Of 6th European Conference on Case Based Reasoning (ECCBR-02).

- [Schonfeld E., 2004]. Don't Just Sit There. Do Something. Business 2.0, 2000.

- [Staab 5. and Studer R., 2004]. Handbook on Ontologies. Springer.

- [Zimmerman J., Kurapati K., Buczak A. L., Schaffer D., Gutta 5. and Martino J., 2004]. TV Personalization System: Design of a TV Show Recommender Engine and Interface. Personalized Digital Television: Targeting Pro grams to Individual Viewers.

Anexo A - Gestión del proyecto

Organización del proyecto

La organización del proyecto se define indicando la estructura organizativa, los límites organizativos y las relaciones existentes entre los diferentes participantes en el proyecto.

Roles organizativos y responsabilidades

El proyecto será realizado por un equipo de seis profesionales que cuentan con una amplia experiencia en el desarrollo de sistemas informáticos. Cada miembro del equipo de desarrollo participará activamente en cada una de las fases del proyecto.

Los distintos roles que participan en el proyecto son:

- **Jefe de Proyecto**: Se encargará de coordinar y dirigir a todo su equipo de trabajo, buscando soluciones adecuadas a los posibles problemas planteados durante el desarrollo del proyecto y asegurando que se cumplen los plazos previstos. Es la persona que se responsabilizará del trato con el cliente, siendo el máximo responsable del proyecto.
- **Analista**: Se responsabilizará del resultado del análisis de la aplicación. Concretará reuniones con el cliente y los usuarios potenciales para captar todos los posibles requisitos y será la persona responsable de la realización del diseño arquitectónico.
- **Responsable de Gestión de Configuración**: Será el responsable de administrar todos los cambios que pueda sufrir cada uno de los productos de los que se compone el proyecto, en cada una de las diferentes fases del ciclo de vida del mismo. Es el responsable de diseñar las políticas a seguir cuando se produce un cambio en cualquier punto del proyecto, ya sea un documento, un fichero de código fuente, etcétera.
- **Responsable de Calidad**: Se encargará de tomar decisiones sobre como aprobar o rechazar los cambios realizados en el proyecto a lo largo del ciclo de vida, asegurándose que en todo momento se cumplen todos los requisitos del sistema, obteniendo así un producto final lo más satisfactorio posible para el cliente.
- **Responsable de Pruebas:** Será el encargado de definir un plan de pruebas y realizar las mismas para comprobar el correcto funcionamiento de la

aplicación, lo que incluirá en todo momento la comprobación y revisión de los documentos generados durante todo el ciclo de vida del proyecto.

- **Técnico informático:** Será el encargado del mantenimiento del hardware, redes y los sistemas operativos de la empresa, necesarios para el desarrollo del proyecto.
- **Programador:** Se encargará de la implementación del sistema informático.

La estructura del equipo de trabajo es jerárquica, siendo el jefe de proyecto el máximo responsable de todas las fases y dejando por debajo de él a los responsables de cada área.

Las tareas que realizará cada una de las personas involucradas en el proyecto son las que corresponden a su respectivo rol asignado, especificado en el apartado anterior.

Nuestra organización tiene la capacidad de adaptarse a las diferentes necesidades del proyecto, de tal manera que todos y cada uno de los componentes de este proyecto serán capaces de hacer frente a cualquier situación imprevista durante todo el ciclo de vida del mismo.

La estructura organizativa que se seguirá en la realización del proyecto, es la proporcionada por la metodología de la E.S.A, en concreto en su versión reducida E.S.A Lite.

Modelo de proceso

El ciclo de vida seguido para el desarrollo del proyecto es el de cascada o waterfall. Este ciclo de vida se caracteriza por que las fases del proyecto son consecutivas y no se solapan, es decir, para comenzar una fase habrá que terminar previamente anterior.

Un caso aparte será la fase de desarrollo que en nuestro caso se hará en paralelo al resto de fases, debido a que es la fase más costosa y los plazos de entrega son muy reducidos. En la figura siguiente podemos apreciar cual es la estructura clásica de un desarrollo en cascada. Posteriormente en el diagrama de Gantt se observará cual es la adaptación para el modelo de cascada que se ha realizado para este proyecto.

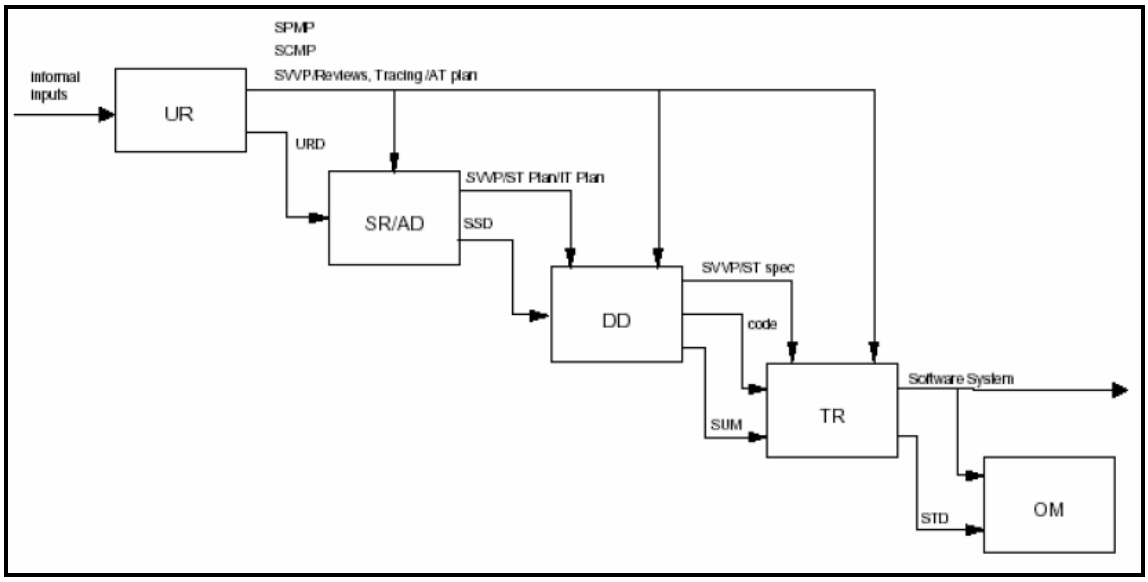


Figura 31 Ciclo de vida

Este ciclo de vida se compone de las siguientes fases:

- UR: Fase de Requisitos de Usuario
    - Fase de definición del problema, en la cual los requisitos del usuario son capturados y se define el ámbito del sistema.
- SR/AD: Combinación de las fases de Requisitos Software y Diseño Arquitectónico
    - Consiste en una fase de análisis y solución del sistema
- DD: Fase de Diseño Detallado
    - Fase de implementación
- TR: Fase de transferencia
    - Fase de entrega al usuario
- OM: Fase de explotación y mantenimiento que en nuestro caso no se llevará a cabo.
    - Esta fase no será llevada a cabo en el desarrollo de este proyecto.

Además existen cuatro actividades de gestión que abarcan todo el ciclo de vida del proyecto software:

- Gestión del proyecto: Recoge el proceso de organización, planificación, control y seguimiento del proyecto software.
- Gestión de la configuración: Proceso de control, identificación y organización de los cambios del software.
- Gestión de la calidad: Proceso de aseguramiento de que los productos y procedimientos están de acuerdo a los estándares y planes definidos.
- Verificación y validación: Proceso de chequeo del software contra sus especificaciones.

Como mencionamos al principio del proyecto, la fase de implementación se realizará durante todo el ciclo de vida del software, hasta la entrega del prototipo.

Informe de Construcción

En este capítulo se especifica el proceso de desarrollo del software, así como otros aspectos relacionados con el proceso de construcción.

Se da por finalizada y válida la fase de construcción, cuando el software ha superado todos los tipos de pruebas que se han definido, respeta el diseño realizado del mismo y, por supuesto, el software hace lo que debe hacer (cumple las funcionalidades para las que fue programado). Además se debe comprobar la instalación del mismo en diferentes entornos.

Para el correcto desarrollo del software se ha respetado, en la medida de lo posible, la planificación y la asignación de tareas realizada.

Para la realización del proyecto, se poseen los siguientes equipos informáticos de última generación.

Equipo	Número
Equipos personales HP pavilon Althon 64 x2 4200+ con 3 GB de memoria	7
Notebooks Acer Aspire 5102WLMI Turion 64 X2 con 1 GB de memoria	2

Tabla 46 - Tabla recursos técnicos.

Además se dispone del siguiente software instalado en los equipos.

Software	Comentario
Microsoft Office 2003	En los equipos personales y portátiles
SO Windows XP Profesional	En los equipos personales y portátiles
Eclipse 3.2.1	En los equipos personales y portátiles
Netbeans 5.0	En los equipos personales y portátiles
JDK 1.5	En los equipos personales y portátiles

Tabla 47 - Tabla recursos software.

Además han sido necesarias las siguientes herramientas:

- Herramientas de análisis y diseño:
  - Microsoft Visio
- Herramientas de diseño gráfico:
  - Corel Draw
- Entorno de desarrollo:
  - Eclipse
- Compilador y Debugger:
  - Java1.5.0 (JVM)

El tiempo empleado por cada rol del equipo para la construcción del sistema queda recogido en las siguientes tablas:

Actividades	Jefe de proyecto	Analista	Resp. Configuración	Resp Calidad	Resp Pruebas	Técnico Informático	Total horas /actividad
Análisis y estudio de la tecnología	0	0	14.4	0	28.8	28.8	72
Desarrollo del framework del proyecto	48.8	34.4	0	28.8	0	0	72

Implementación del Sistema	55.2	76.8	16.8	25.2	42	42	168
Pruebas, correcciones e implantación	19.2	0	4.8	0	14.4	9.6	48
TOTAL	73.2	31.2	36	54	85.2	80.4	490

Tabla 48 - Tabla tiempos roles.

Anexo B - Manual de usuario

Instalación

En este capítulo se describe el proceso de instalación y ejecución del sistema.

El sistema se presentará al usuario en un CD-ROM. En dicho CD-ROM se incluirá tanto el código fuente como el código compilado.

Para instalar el software, es necesario que el equipo posea una versión de la máquina virtual de Java igual o superior a las 1.5 y una red con conexión a Internet si se quiere distribuir las aplicaciones en varios equipos.

Se puede descargar el la máquina virtual desde http://java.sun.com/javase/downloads/index_jdk5.jsp. Si lo desea, puede descargar el entorno de desarrollo completo. Puede instalarse una versión superior a la 1.5.

El proceso de instalación de JDK 1.5 es muy sencillo y es prácticamente autocontenido. Una vez descargado el ejecutable pulse doble clic y le aparecerá una pantalla similar a esta:

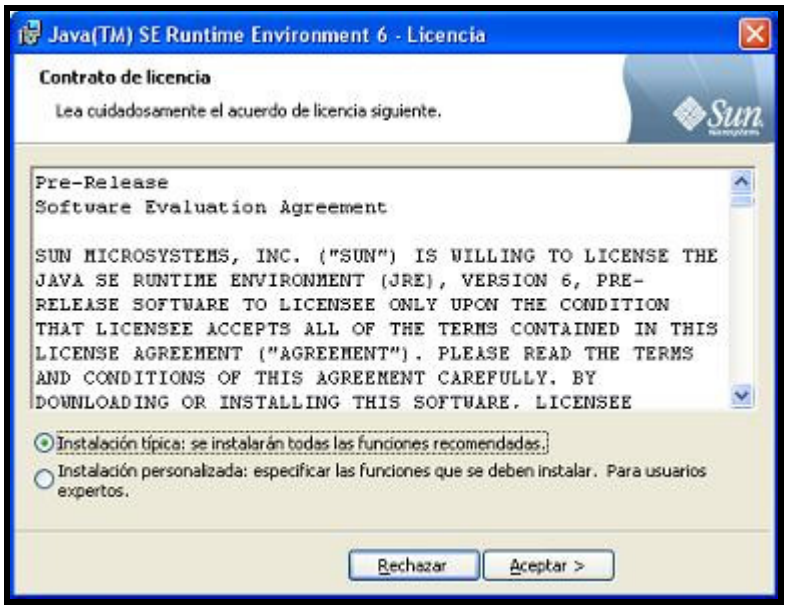


Figura 32 Instalación Java I.

Se deberá leer el contrato de licencia y aceptarlo si estamos de acuerdo. A continuación se marcará la opción instalación típica y se pulsará aceptar.

A continuación se instalaran en el sistema los archivos que java necesita.

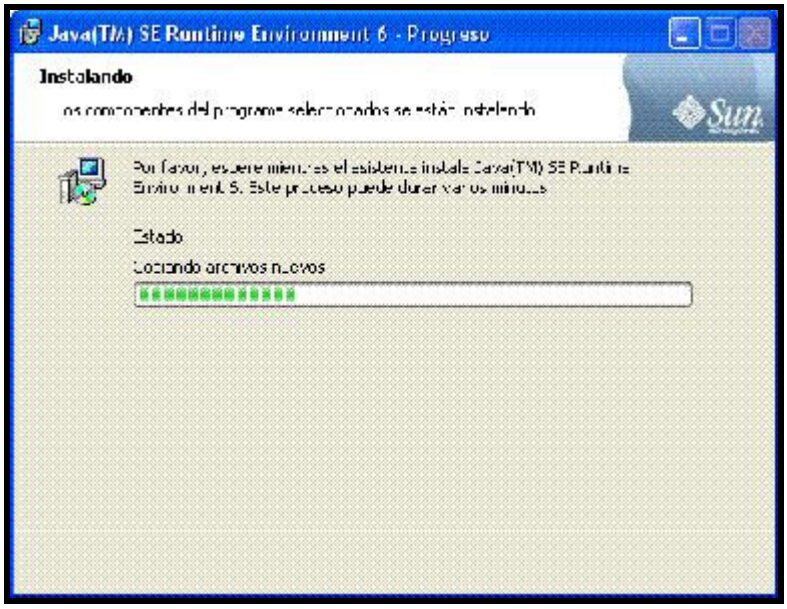


Figura 33 Instalación Java II.

Una vez finalizado se deberá comprobar que la configuración de las variables de entorno del sistema es la correcta. Para acceder a las variables de entorno vaya a MI PC, haga clic con el botón derecho para desplegar el menú y seleccione *propiedades*. Le aparecerá una pantalla como la siguiente:

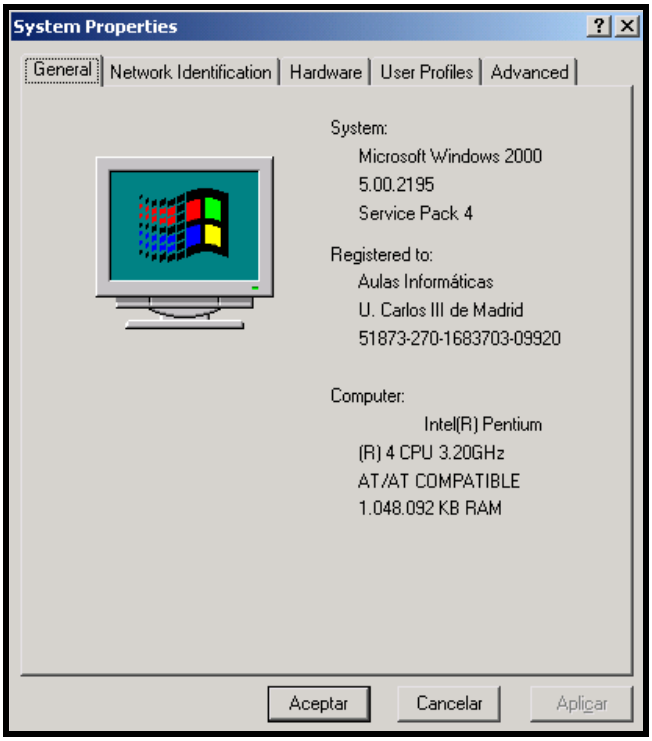


Figura 34 Propiedades del sistema.

A continuación pulse la pestaña *avanzado* y seleccione el botón del centro denominado variables de entorno. En dichas variables de entorno debe editar lo siguiente:

Si no ha definido en la variable PATH la ubicación de la máquina virtual de java (normalmente en la carpeta "bin" de la carpeta de instalación de java), debe pulsar el botón *editar* y añadir la ruta de java con un ; entre la última que hubiese y la ruta de java, así como en la variable CLASSPATH debe introducir tanto la carpeta donde se encuentran las librerías de java (Normalmente la carpeta "lib" dentro de la carpeta de instalación de java) como la carpeta "."que representa al directorio actual, para que encuentre las clases de la aplicación. Ambas rutas deben ser introducidas con un ; de separación. Si no existiese la variable CLASSPATH debe crearla pulsando el botón *nueva.* Con esto tendrá configurado su entorno Java.

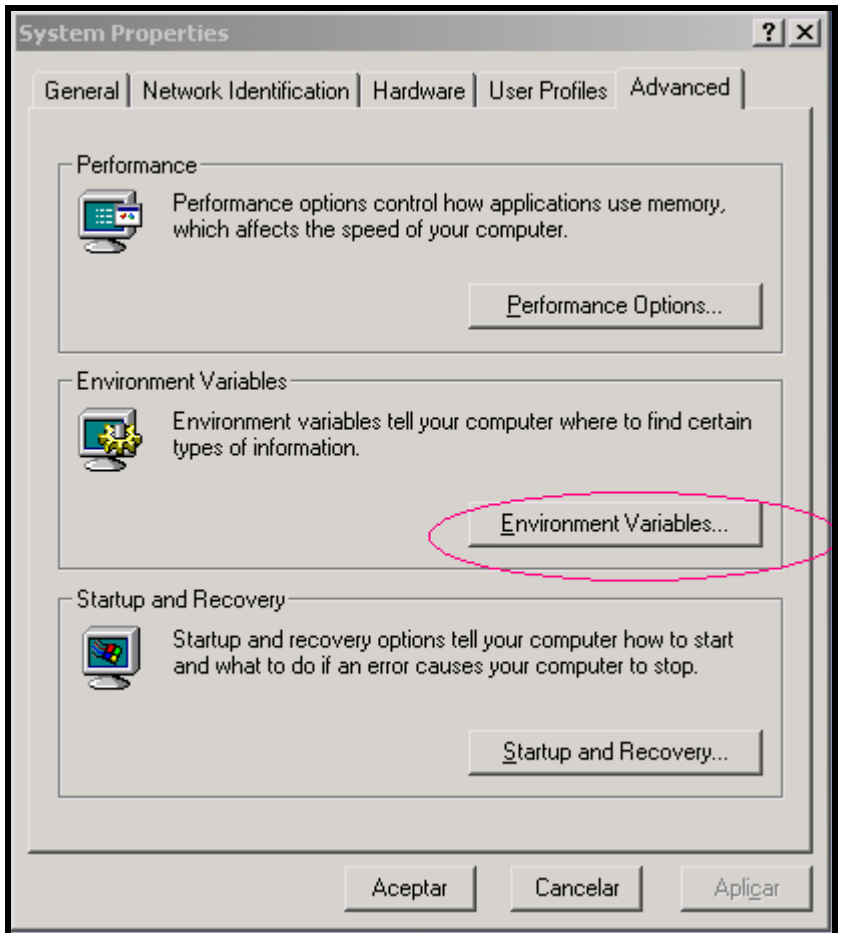


Figura 35 Variables del sistema.

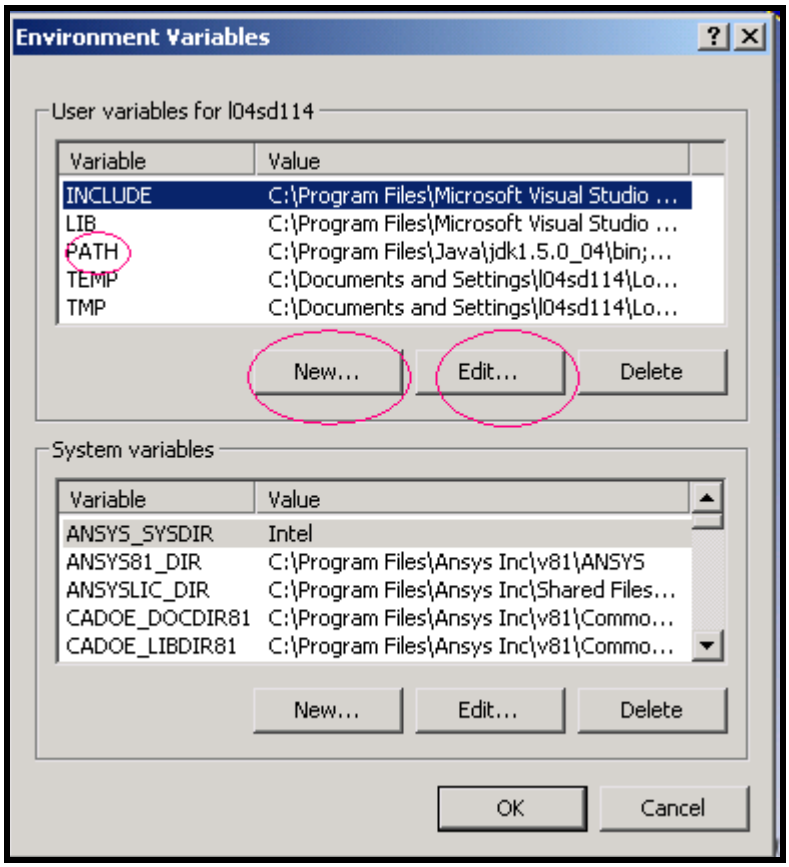


Figura 36 Configuración classpath.

Al ser un programa escrito en java, puede instalarse sobre cualquier sistema operativo y no necesita de ningún proceso de instalación salvo la configuración de las variables de entorno. Si su sistema ya estaba configurado previamente o lo acaba de configurar, la instalación de la aplicación es muy sencilla. Tanto el programa como el código vienen comprimidos en formato zip. Debe descomprimirse todo el contenido en el directorio de trabajo de la aplicación. Una vez configuradas las variables d entorno e instalado correctamente Java 1.5.

Fases de la ejecución del sistema

En este punto vamos a explicar cada uno de los pasos que forman el proceso completo de la recomendación para un usuario, con la aplicación visual que se ha realizado.

Nada más ejecutar la aplicación, se nos mostrará una interfaz que simulará la adaptación del programa visualizado en nuestra televisión.

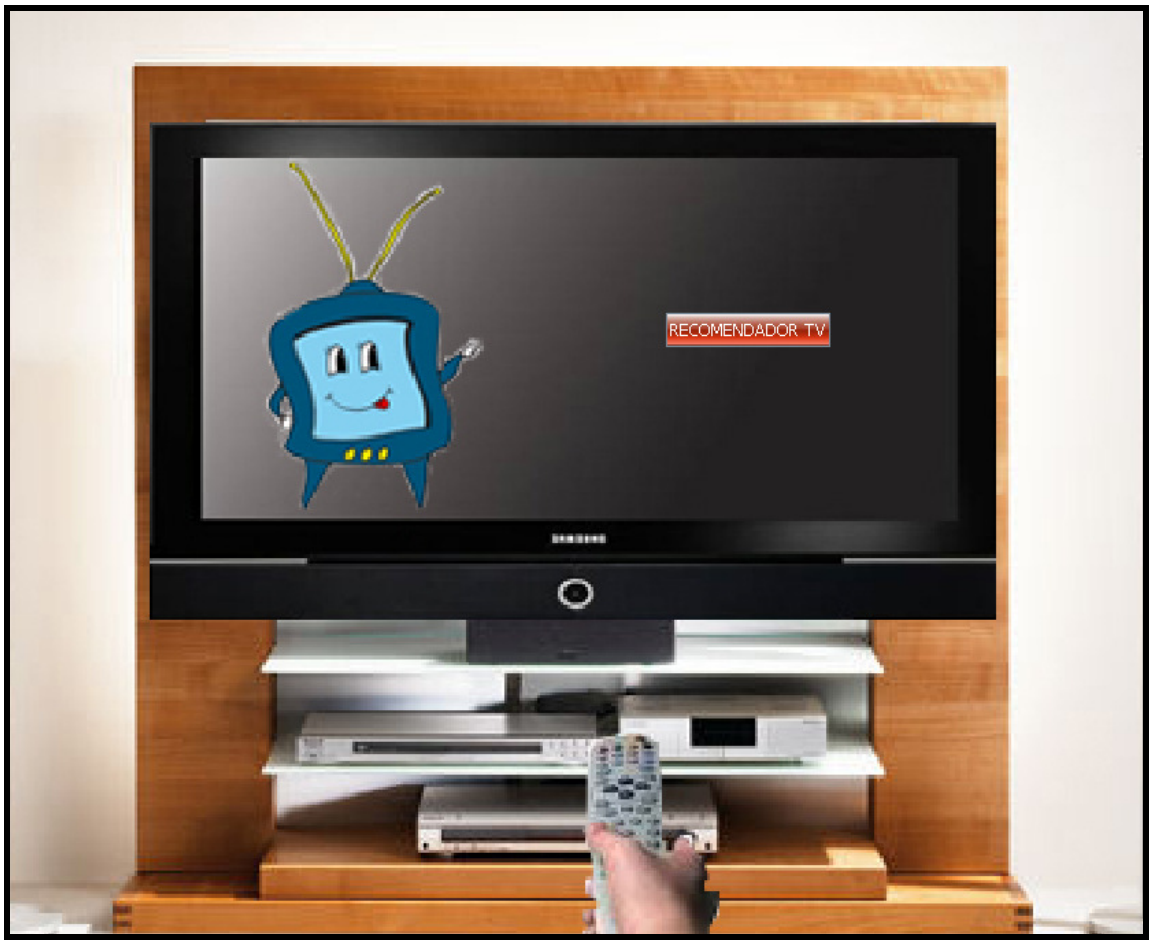


Figura 37 Pantalla inicial proceso recomendación.

Si pulsamos en la opción RECOMENDADOR TV, se nos presentará una ventana emergente, donde indicaremos cual ES id del usuario para el que queremos realizar la simulación, mediante un fichero xml, así como el algoritmo de comparación de keywords, bien LIKE, o bien EQUALS.

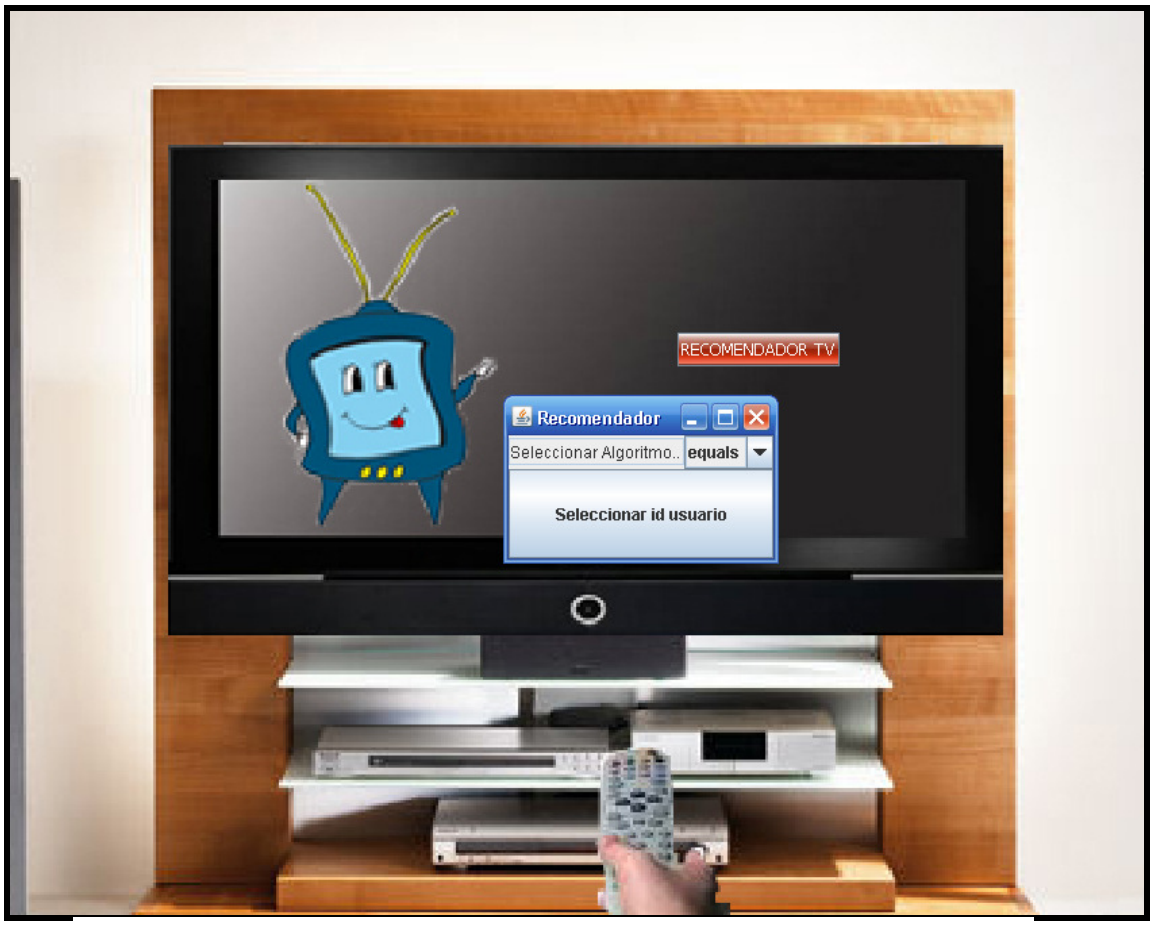


Figura 38 Selección id usuario.

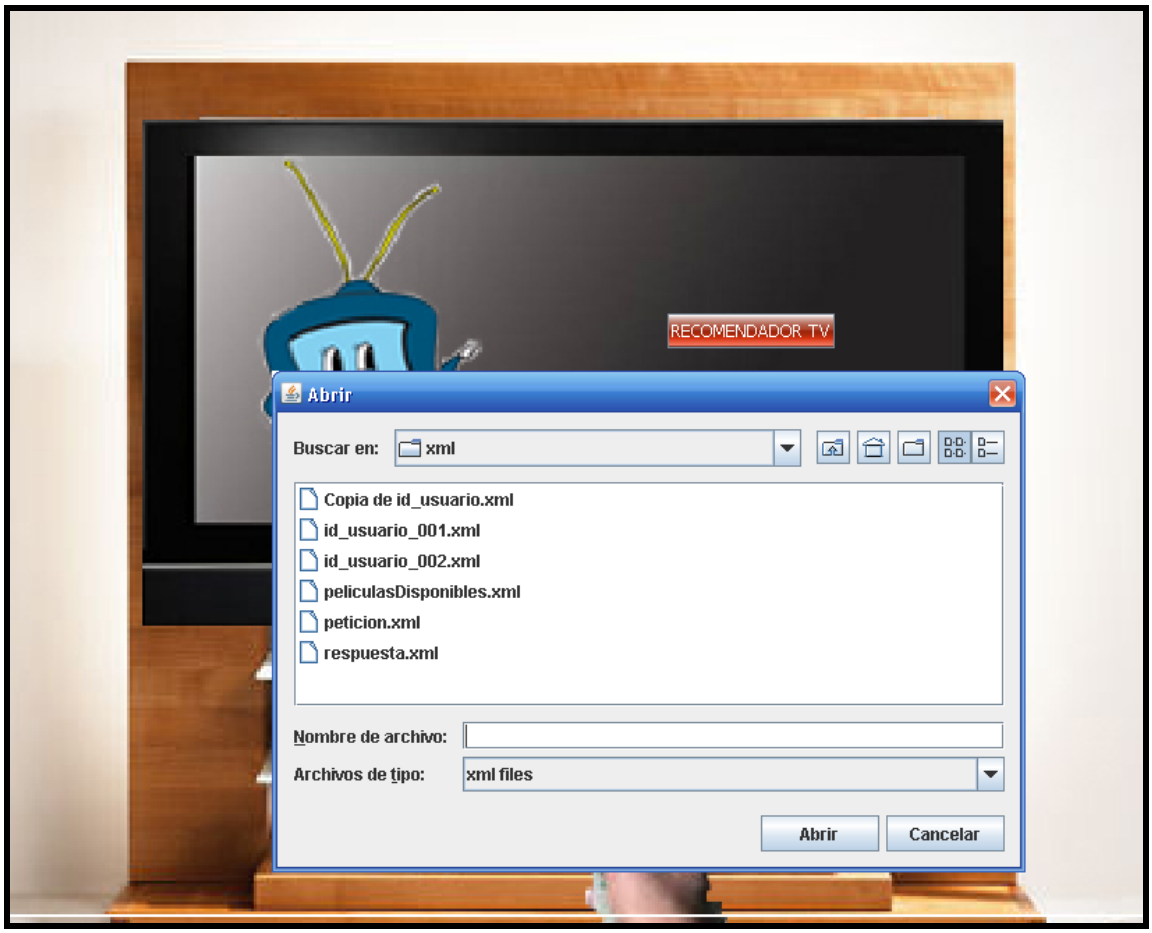


Figura 39 Selección xml usuario.

Si por ejemplo, elegimos el fichero id_usuario_001.xml, la aplicación tomaría como petición, la correspondiente a este id de usuario, y la aplicación ya estaría preparada para realizar la simulación. Sólo tendríamos que pulsar el botón Start para empezar la fase 1 de la simulación.

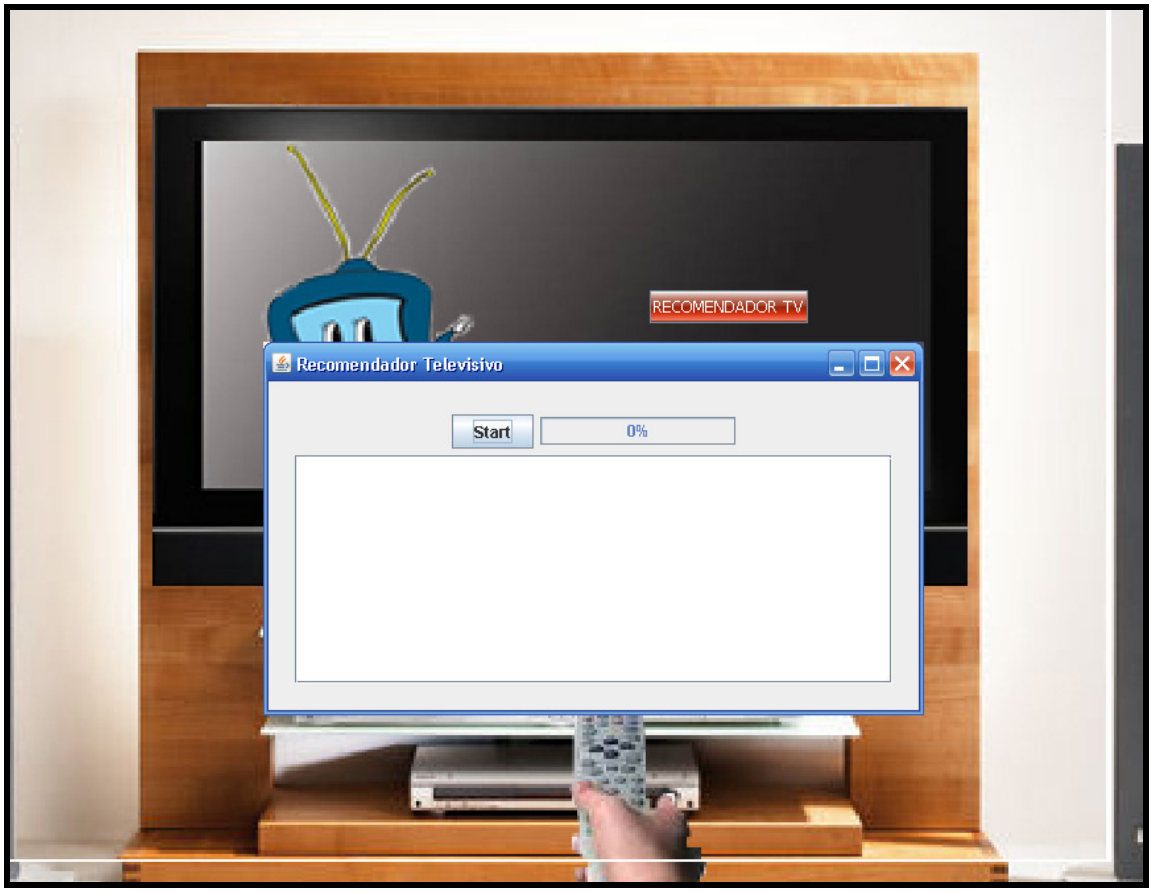


Figura 40 Inicio Proceso 1 recomendación.

La interfaz nos va mostrando en cada una de las tareas o fases, el porcentaje del trabajo realizado, mediante una barra de progreso. Cuando la tarea finaliza, nos informa de ello, y del comienzo del siguiente proceso. En la siguiente figura nos informa de que la tarea 1 “Carga de keywords del usuario” ha finalizado.

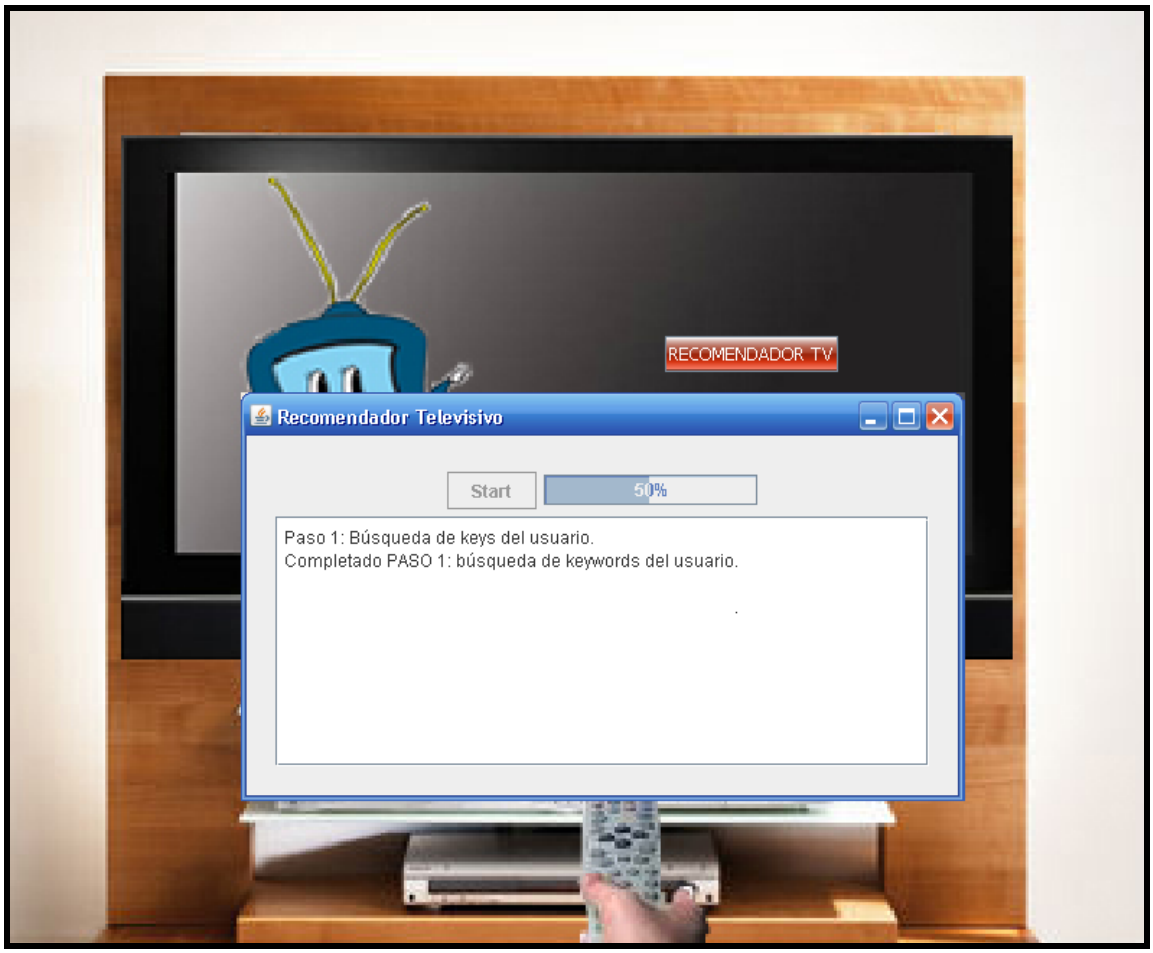


Figura 41 Fin proceso 1 de recomendación.

En la siguiente figura nos informa de que la tarea 2 “Carga de sinónimos” ha finalizado y comenzará con el proceso 3.

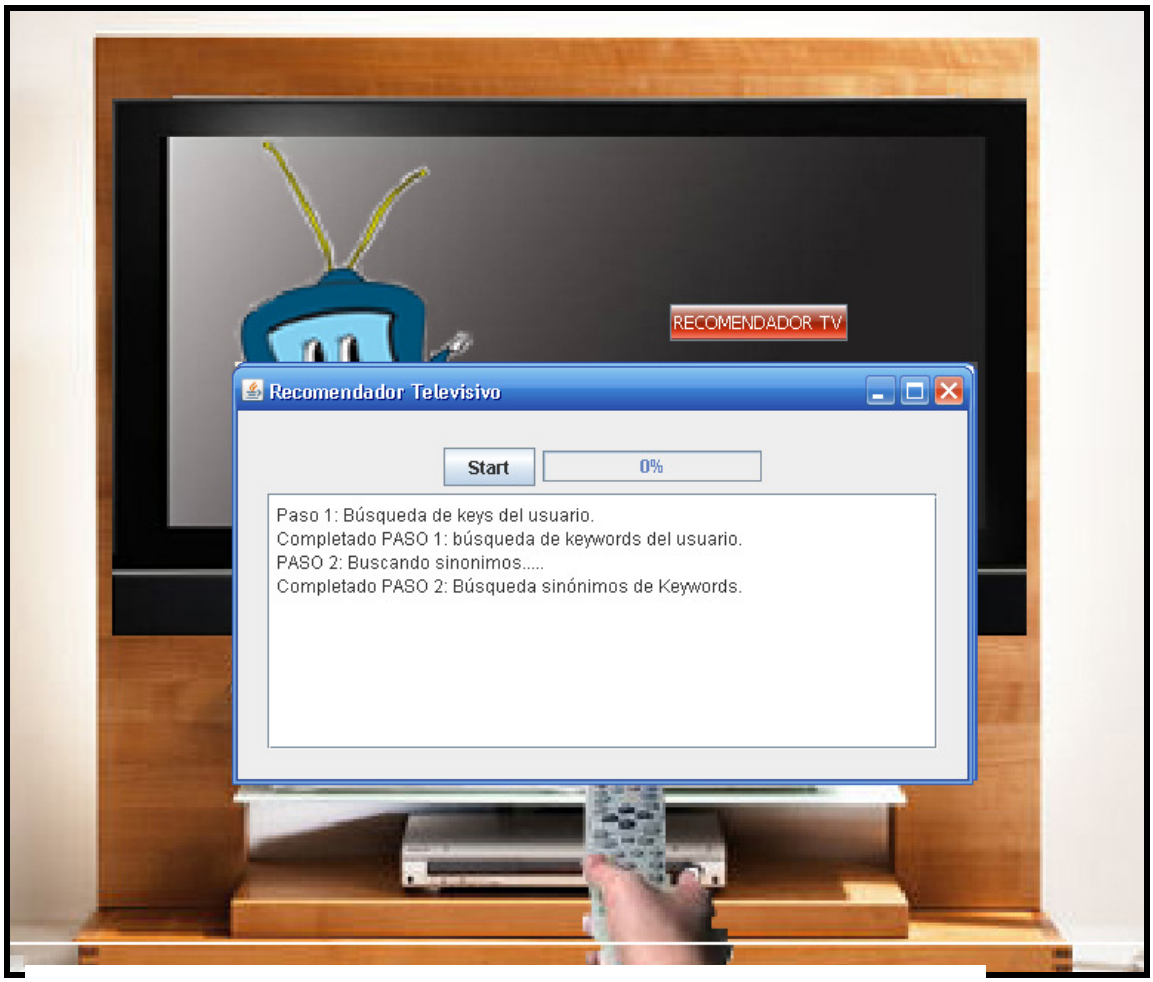


Figura 42 Fin proceso de recomendación 2

En la siguiente figura nos informa de que la tarea 3 "Asignación de pesos de películas" ha finalizado, y la aplicación seguirá ejecutando el siguiente proceso.

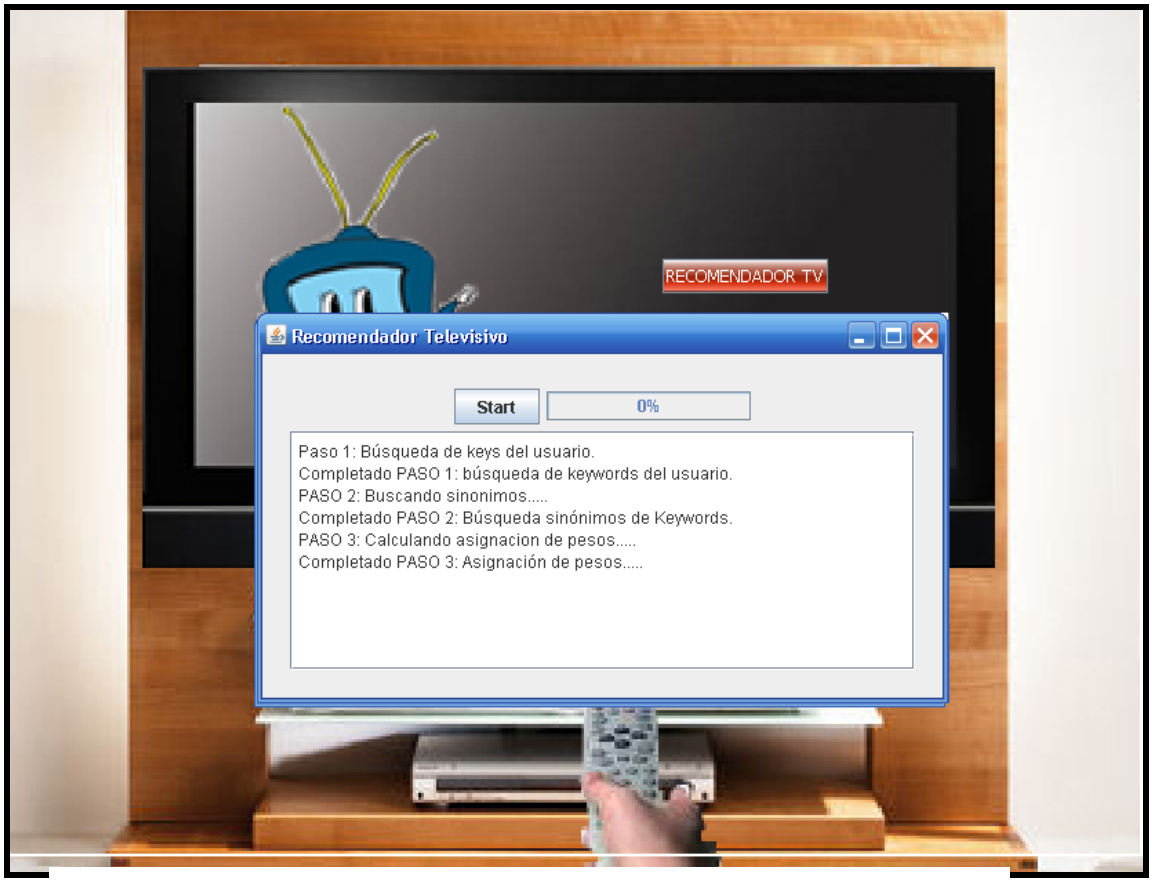


Figura 43 Fin proceso de recomendación 3

En el siguiente paso, la aplicación creará un fichero xml, con toda la información de las películas recomendadas, ordenada por afinidad.

A continuación podemos ver un ejemplo del fichero final con la recomendación televisiva, en el que vienen ordenadas de 1 a 15 todas las películas recomendadas para el usuario, con un número que identifica de manera unívoca a la película.

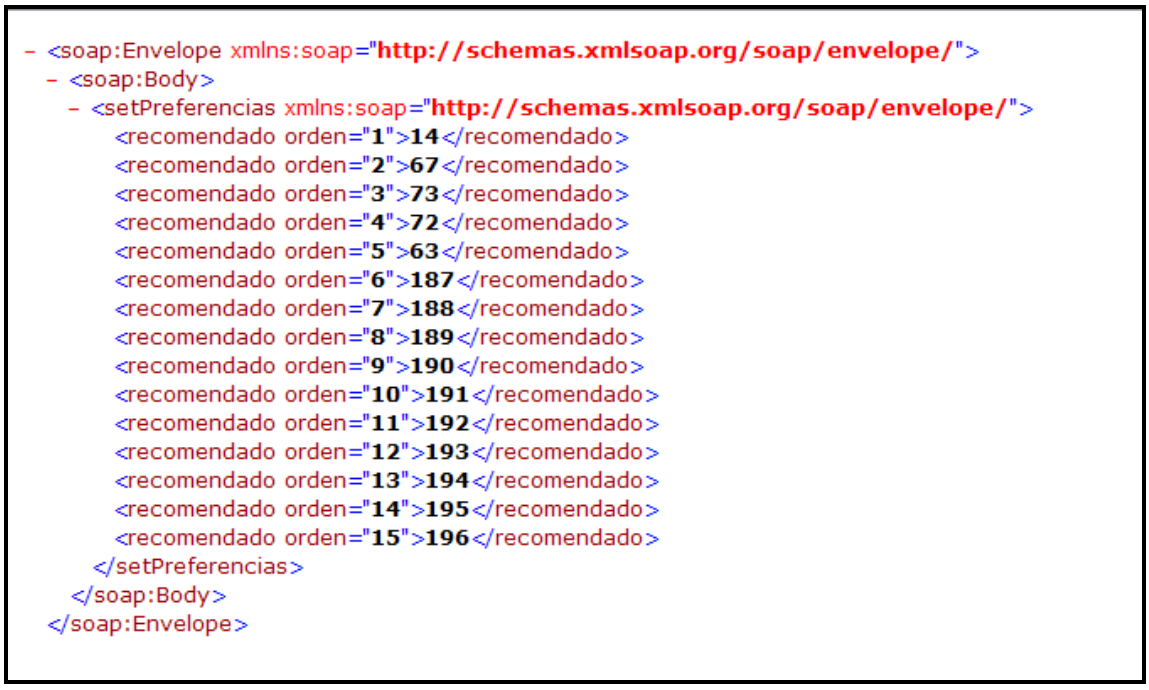


Figura 44 Fichero xml final recomendación.

El siguiente paso que realizará la aplicación será crear otro fichero xml, similar al anterior, pero incorporando el título de la película en lugar del código unívoco de ésta, para poder presentar la información de una manera más limpia e intuitiva.

A continuación se muestra un ejemplo del fichero final que creará la aplicación y será enviado al decodificador para poder presentar al usuario final.

Figura 45 Fichero xml final adaptado recomendación.